SLIM: Subtrajectory-Level Elimination for More Effective Reasoning

Xifeng Yao[*], **Chengyuan Ma**[*], **Dongyu Lang**[*], **Yinhao Ni, Zhiwei Xu, Huarui Xie, Zihao Chen, Guang Shen, Dandan Tu, Yi Bai, Changzheng Zhang**[†]
Huawei Technologies Co., Ltd.
{yaoxifeng, machengyuan1, zhangzhangzheng}@huawei.com
langdongyu@h-partners.com

Abstract

In recent months, substantial progress has been made in complex reasoning of Large Language Models (LLMs), particularly through the application of test-time scaling. Notable examples include, though are not limited to, OpenAI's o1/o3/o4 series and DeepSeek-R1. When responding to a query, these models generate an extended reasoning trajectory, during which the model explores, reflects, backtracks, and self-verifies before arriving at a conclusion. However, fine-tuning models with such reasoning trajectories may not always be optimal. Our findings indicate that not all components within these reasoning trajectories contribute positively to the reasoning process; in fact, some components may affect the overall performance negatively. In this study, we divide a reasoning trajectory into individual subtrajectories and develop a "5+2" framework to: (1) systematically identify suboptimal subtrajectories within the reasoning trajectory based on five human-established criteria; (2) assess the independence of the suboptimal subtrajectories identified in (1) from the subsequent content, ensuring that their elimination does not compromise overall flow and coherence of the reasoning process. Additionally, a sampling algorithm, built upon the "5+2" framework, is employed to select data whose reasoning process is free from suboptimal subtrajectories to the highest degree. Experimental results demonstrate that our method can reduce the number of suboptimal subtrajectories by 25.9% during the inference. Furthermore, our method achieves an average accuracy of 58.92% on highly challenging AIME24, AIME25, AMC24 and MATH500 benchmarks with only two thirds of training data, surpassing the average accuracy of 58.06% achieved with the entire data, and outperforming open-source datasets, including s1K-1.1, Light-R1-SFT-stage-1, OpenR1-Math-94k, and OpenThoughts-114k, when fine-tuning Qwen2.5-Math-7B. Finally, we have validated the efficacy of our method under resource-constrained scenarios, where it exhibits performance improvements across different maximum inference token limits: 2k, 4k, 8k, and 16k tokens.

1 Introduction

Large language models (LLMs) have been rapidly evolving in their ability to tackle complex reasoning tasks. Recently, in the domain of LLMs, Reinforcement Learning (RL) employing an outcome-based reward has attracted public attention, as it grants the model extensive freedom to explore, reflect, backtrack, and self-verify, a process known as test-time scaling (DeepSeek-AI et al., 2025; Luo et al., 2025). RL-ed LLMs, exemplified by DeepSeek-R1 (DeepSeek-AI et al., 2025), have demonstrated robust capabilities in handling complex reasoning tasks, and are consequently often used as teacher models in knowledge distillation, enhancing the reasoning capabilities in other models or cold-starting them with a test-time scaled output format through Supervised Fine-Tuning (SFT) (DeepSeek-AI et al., 2025; Yang et al., 2025; Wen et al., 2025). However, reponses generated by RL-ed LLMs do not always guarantee the highest quality, as the unconstrained freedom during the RL training can introduce inefficiencies or counterproductive elements, such as prematurely abandoned steps or repetitive verifications, even within correct solutions, which will be illustrated in Section 3.1. Fine-tuning a model using such solutions would be suboptimal, as they could potentially decrease both the model's accuracy (Ye et al., 2025) and thinking efficacy (Wang et al., 2025).

Naturally, the following question arises: given a set of QA pairs, where the answers are distilled from a RL-ed LLM, how can we select the QA pairs that are free from these inefficiencies and counterproductive elements to fine-tune another

[*] Co-first authors
[†] Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 3377–3394

model? To address this question, we first recall that answers from a RL-ed LLM, such as DeepSeek-R1, typically take the form in Appendix A.

For a QA pair, we divide its thinking process into individual approaches, referred to as **subtrajectories** in later discussions. We propose five criteria: *Effort*, *Effectiveness*, *Coherence*, *Preliminary Conclusion*, *Valid Verification*, to assess each subtrajectory, determining whether it contributes positively to problem-solving from a specific perspective. If a subtrajectory fails to meet a criterion, we will further assess its independence within the thinking process and determine whether it can be removed without impacting the understanding and coherence of subsequent reasoning process. After eliminating suboptimal and independent subtrajectories, we will assign a quality score to the QA pair: first, assign a score to each subtrajectory based on the five criteria; second, aggregate these scores with weights proportional to the number of tokens in each subtrajectory.

In addition to data quality, our analysis reveals that the distribution of the number of subtrajectories within the dataset also influences the model's reasoning ability. Accordingly, following the modification and computation of the quality scores, we develop a sampling algorithm of selecting QA pairs for supervised fine-tuning. This algorithm considers both the quality scores and the number of subtrajectories in the thinking process, achieving a balance through weights determined by the Kullback-Leibler (KL) divergence (Joyce, 2011) between the distribution of number of subtrajectories in the entire dataset and that in the sampled dataset. This approach enables the selection of efficient and productive QA pairs based on an in-depth assessment of their thinking processes, while preventing the algorithm from disproportionately favoring thinking process with fewer subtrajectories.

Comprehensive experimental results illustrate that our methods achieve an average accuracy of 58.92% on the highly challenging AIME24, AIME25, AMC24, and MATH benchmarks, utilizing merely two-thirds of the curated training data. This performance surpasses the 58.06% accuracy obtained with the full dataset. Meanwhile, the number of suboptimal subtrajectories decreases by 25.9% during the inference, which suggests a more profound and efficient reasoning paradigm. The concurrent enhancement in accuracy and thinking efficacy underscores that the responses generated from RL-ed LLMs indeed exhibit significant quality issues, and our data quality pipeline, encompassing suboptimal subtrajectory elimination and sampling strategy, demonstrates a robust capability in mitigating these issues.

In summary, our contributions are: (1) We propose a "5+2" framework to assess and modify the thinking processes generated from RL-ed LLMs at subtrajectory level. (2) We develop a sampling algorithm aimed at selecting efficient and productive QA pairs for supervised fine-tuning based on subtrajectory assessment. (3) We conduct comprehensive experiments and ablation studies to illustrate the effectiveness of '5+2" framework and samping strategy, which enhance both model accuracy and thinking efficacy.

2 Data Curation

In this section, we discuss the process of constructing OpenSourceR1-Hard and DeepMath-Hard, the source datasets that we use for our subsequent studies. It should be noted that our hypotheses and methodologies in section 3 are both formulated and validated using OpenSourceR1-Hard, and the DeepMath-Hard dataset is regarded as an out-of-distribution test set. Both datasets undergo the following filtering processes, including basic quality filtering and difficulty filtering. The construction and filtration processes of the two datasets are elaborately detailed in Appendix B. We also decontaminate the collected dataset against the evaluation benchmarks mentioned in 4.1 using 15-grams.

3 Sampling at the Subtrajectory Level

3.1 A Deep Dive into Subtrajectories

When responding to a query, DeepSeek-R1, along with several RL-ed LLMs, initiates a thinking process. During this process, the model explores multiple approaches (Qin et al., 2024), reflects on the current approach, reverts to previous steps when the current approach is longer deemed viable, and conducts self-verification. The attempted approaches, hereafter referred to as **subtrajectories**, are demarcated clearly, typically initiating with phrases such as "Alternatively", "Another method", and similar expressions. However, the quality of these subtrajectories is inconsistent, which in turn impacts the overall quality of the thinking process. After reviewing dozens of thinkings from OpenSourceR1-Hard, we identify that low-quality subtrajectories frequently manifest in the following forms (see Appendix C for examples):

1. *The subtrajectory proposes a method without attempting it.*

2. *The subtrajectory attempts to solve the problem in an ineffective manner.*

3. *The subtrajectory has logical discontinuities.*

4. *The subtrajectory transitions to the next one without reaching any conclusions.*

5. *The subtrajectory contains redundant self-verification(s).*

We prompted QwQ-32B (Team, 2025) to assess whether subtrajectories in the OpenSourceR1-Hard dataset exhibit any of the aforementioned issues. The evaluation result revealed that 50.16% of all subtrajectories contain at least one of the five defined low-quality characteristics.

3.2 Identifying and Eliminating Suboptimal Subtrajectories

To identify the five inefficient and counterproductive components, we have established five specific criteria. We prompt QwQ-32B with these criteria to evaluate each subtrajectory. These five criteria form the "+5" component of our "5+2" framework.

1. *Effort*: The subtrajectory should not only introduce a method but also demonstrate its relevance to the current context. This involves providing a detailed explanation of the method and then applying it to address the problem at hand, integrating it with the preceding discussion or the problem statement.

2. *Effectiveness*: The subtrajectory should attempt the problem in an effective manner. This may involve: simplifying the problem, refining previously suggested steps, advancing the problem-solving process, clarifying the limitations of the applied methods, or substantiating earlier conclusions.

3. *Coherence*: Each step within the subtrajectory is logically connected, ensuring no logical leaps occur in the reasoning process. Every intermediate result must be derived through computation or rigorous proof.

4. *Preliminary Conclusion*: Before transitioning to the next subtrajectory, this subtrajectory should draw a preliminary conclusion, which may include a final answer, intermediate findings, an evaluation of the current approach, or suggestions of other viable approachs.

5. *Valid Verification*: The subtrajectory avoids repetitive verification of the same statement using the identical method, and it does not re-verify statements that have been verified in previous subtrajectories.

If a subtrajectory fails to meet any of the five criteria, it is classified as a **suboptimal subtrajectory**. The existence of suboptimal subtrajectories can degrade the overall quality of the thinking process. However, discarding QA pairs that include any suboptimal subtrajectory would significantly reduce the data size available for supervised fine-tuning. Instead, we opt to eliminate any identified suboptimal subtrajectory within the thinking process utilizing the five criteria.

When eliminating subtrajectories, it is crucial to maintain the overall flow and structure of the thinking process. An example of non-eliminable suboptimal subtrajectory is shown in Appendix D. Note that the first subtrajectory in the example fails to attempt the approach it proposes, thus violating the first criterion. However, this subtrajectory cannot be eliminated because the area fomula derived in it is revisited in the third subtrajectory, in which a valid attempt is made. Given this dependency, the first subtrajectory must be retained.

Therefore, subtrajectories that are suboptimal should not be removed if their removal impairs the understanding of the subsequent content. To be more precise, upon identifying a suboptimal subtrajectory, we will prompt QwQ-32B to evaluate its independence from subsequent subtrajectories. Should this suboptimal subtrajectory be determined to be independent, it will be subject to elimination.

1. *Independence*: Assessing whether the parameters, variables, algebraic expressions, conclusions, or verifications defined in the current subtrajectory are used in later content.

2. *Elimination*: If the current subtrajectory is relied upon by subsequent content, it should be retained. Conversely, if a subtrajectory is suboptimal and independent of subsequent subtrajectories, it should be eliminated.

This independence assessment and elimination mechanism constitutes the "+2" component of our "5+2" framework.

3.3 The Sampling Algorithm

3.3.1 Scoring a Thinking Process

Due to the existence of suboptimal subtrajectories that cannot be eliminated, the **revised thinking process**, i.e., thinking process after elimination of independent suboptimal subtrajectories, cannot be problem-free. Therefore, we introduce a scoring mechanism to assess the quality of the revised thinking process, in accordance with the five criteria outlined in the preceding section. This scoring mechanism will be instrumental in the selection of QA pairs for supervised fine-tuning.

Given a QA pair, we extract its thinking process. Next, we prompt QwQ-32B to evaluate each subtrajectory within this thinking process against the five criteria, with the aim of identifying and eliminating those suboptimal subtrajectories that are independent, as detailed in Section 3.2. Each of the remaining subtrajectories is awarded $\frac{1}{5}$ points for each of the five criteria it satisfies:

$$\begin{aligned} &Score(\text{subtrajectory}) := \\ &\sum_{j=1}^{5} \frac{1}{5} \cdot \mathbb{1}\Big[\text{subtrajectory satisfies criterion}_j\Big]. \end{aligned} \tag{1}$$

Note that a score ranging from 0 to 1 is assigned to each subtrajectory. We will aggregate these individual scores into a single score that accurately reflects the overall quality of the thinking process.

3.3.2 Varied Weights Based on Token Counts

The length of subtrajectories is a critical factor. For longer suboptimal subtrajectories in the revised thinking process, a larger penalty should be imposed in contrast to their shorter counterparts. Consequently, when aggregating the scores of each subtrajectory, we apply a weight that is determined by the token count of the respective subtrajectory:

$$\begin{aligned} &QualityScore(\text{thinking}) := \\ &\sum_{i=1}^{n} \frac{T(\text{subtrajectory}_i)}{T(\text{thinking})} \left(Score(\text{subtrajectory})\right), \end{aligned} \tag{2}$$

where n is the number of subtrajectories within the thinking process, and $T(\cdot)$ returns the number of tokens of the input string. The flow chart in Appendix M demonstrates the computation process of the varied weights based on token counts.

Technically speaking, the quality score is specifically defined on the thinking process within the answer of a QA pair. Given that each QA pair contains exactly one thinking process, we will adopt a less rigorous notation: *QualityScore*(QA pair), to denote the quality score of the thinking process within the answer of that QA pair.

3.3.3 Sampling on Quality Score and Distribution of Subtrajectory Counts

Naturally, after calculating the quality score of a thinking process, we can establish a threshold and select QA pairs whose thinking process scored above this threshold. However, we notice that the scoring mechanism disproportionately favors thinking processes with fewer subtrajectories, as they are less prone to violate criterion 1, 2, 4. The findings are detailed in Appendix F.

Theoretically speaking, a SFT dataset comprising an excessive number of QA pairs with an extremely low number of subtrajectories may lead to a reduction in the SFTed model's exploratory ability, confining its search to a limited space and thereby impairing its performance on complex reasoning tasks. Therefore, when sampling based on quality scores, it is essential to introduce a constraint by incorporating a penalty term that reflects the percentage change in the frequency of number of subtrajectories within the thinking process of the sampled dataset and the entire dataset. The detailed sampling algorithm is in Appendix G.

Through the sampling algorithm, we may select QA pairs that are aligned with the five criteria outlined in Section 3.2, while considering the number of subtrajectories as intact as possible.

4 Experimental Results

4.1 Setup

Training: We conduct supervised fine-tuning on Qwen2.5-Math-7B across two datasets: OpenSourceR1-Hard and DeepMath-Hard to evaluate the effectiveness of our methods in the domain of mathematics. The detailed training configurations is in Appendix H.

Evaluation: We assess the effectiveness of our methods using a range of mathematics benchmarks, including AIME24, AIME25, MATH500, AMC24, as detailed in Appendix I. The evaluation methods are detailed in Appendix J.

4.2 Ablation Studies

We conduct ablation studies to assess the efficacy of the "5+2" framework and the sampling algorithm on both our in-distribution

dataset OpenSourceR1-Hard, and our out-of-distribution dataset DeepMath-Hard. Regarding the OpenSourceR1-Hard dataset (around 60k samples), we have curated various fractions of the dataset, including the entire dataset, two-thirds of the dataset, and one-third of the dataset. Regarding the DeepMath-Hard dataset (around 12k samples), we have curated two fractions: the entire dataset and two-thirds of the dataset. Due to the relatively limited size of the DeepMath-Hard dataset, we did not curate a one-third fraction in our analysis.

For each sampled fraction, we consider the following four configurations, as detailed in Appendix K: (1) Elimination with Sampling Algorithm (*E+SA*); (2) No Elimination with Sampling Algorithm (*NE+SA*); (3) Elimination without Sampling Algorithm (*E+NSA*); (4) No Elimination without Sampling Algorithm (*NE+NSA*).

For the entire dataset, only the configurations *E+NSA* and *NE+NSA* are employed, as the sampling algorithm is inapplicable in this context.

The performance of the OpenSourceR1-Hard models is detailed in Table 1. The results demonstrate that the elimination of suboptimal subtrajectories enhances the model's performance across all comparative groups, regardless of the application of the sampling algorithm. Specifically, within the entire dataset, the *E+NSA* configuration achieves an accuracy of 59.60%, outperforming the *NE+NSA* configuration, which attains 58.06%. Similarly, in the two-thirds of the dataset, when the sampling algorithm is applied, *E+SA* achieves an accuracy of 58.92%, representing a 1.86% improvement over *NE+SA*. This enhancement can be attributed to the efficacy in eliminating suboptimal subtrajectories, thereby optimizing the overall solution's efficiency despite a reduction in token length. In the one-third of the dataset, elimination suboptimal subtrajectories achieves approximately the same accuracy as configurations without the elimination process. To our best knowledge, this similarity in performance is partly due to the 7B model's limited math capabilities compared to larger models such as the 32B variant, making its performance highly susceptible to the quantity of data and tokens utilized in the SFT process. Despite eliminating suboptimal subtrajectories further reducing the number of tokens used in SFT, it manages to maintain a comparable level of accuracy to configurations with the original solution.

Moreover, the integration of the "5+2" framework with sampling algorithms demonstrates a pronounced capability in augmenting model performance. Specifically, the implementation of *E+SA* significantly enhances model accuracy from 56.23% (as observed in *NE+NSA*) to 58.92% in the two-third of the dataset. A similar observation has been made within the one-third of the dataset. Additionally, the *E+SA* model in the two-third of the dataset demonstrates a 0.86% better performance compared to the *NE+NSA* model in the entire dataset. This suggests that although reductions in sample size and token amounts can significantly influence a 7B model in SFT process, the "5+2" framework together with the sampling algorithm are particularly effective in identifying optimal QA pairs from the entire dataset, thereby achieving enhanced performance.

Methods	AIME25	AIME24	MATH500	AMC24	Average
Entire Dataset					
E+NSA	35.03	44.15	90.25	68.98	**59.60**
NE+NSA	29.18	47.50	88.90	66.65	58.06
Two-thirds of the Dataset					
E+SA	38.63	39.43	90.55	67.05	**58.92**
NE+SA	35.85	36.70	89.80	65.90	57.06
E+NSA	37.50	38.35	89.40	60.40	56.41
NE+NSA	31.65	35.80	89.25	68.23	56.23
One-third of the Dataset					
E+SA	29.45	35.00	87.20	66.30	**54.49**
NE+SA	30.55	35.00	87.05	65.33	54.48
E+NSA	27.50	34.15	87.95	60.98	52.65
NE+NSA	27.50	33.90	87.90	61.73	52.76

Table 1: OpenSourceR1-Hard: The "5+2" framework and the sampling algorithm performance across mathematical benchmarks

A similar trend is observed on our out-of-distribution dataset DeepMath-Hard, as summarized in Table 2. Specifically, within the entire subset, the *E+NSA* configuration achieved an accuracy of 52.53%, significantly surpassing the 50.21% accuracy of the *NE+NSA* configuration. Moreover, in the two-third of the dataset, the implementation of *E+SA* yielded an accuracy rate of 49.12%, outperforming the 47.05% achieved by *NE+NSA*. These findings indicate that, even when evaluated on an out-of-distribution dataset, the integration of the "5+2" framework and the sampling algorithm exhibits superior performance across various data sizes, outperforming configurations that do not incorporate these methods.

In addition, we conducted two extra sets of ablation studies to validate our methods in Section 3. The first ablation study compares equal weights, detailed in Appendix E, against varied weights. The second one contrasts the presence and absence of the sampling algorithm. The results of these two

Methods	AIME25	AIME24	MATH500	AMC24	Average
Entire Dataset					
E+NSA	29.45	28.60	87.65	64.40	**52.53**
NE+NSA	25.00	28.60	87.40	59.85	50.21
Two-thirds of the Dataset					
E+SA	27.23	27.23	85.20	56.80	**49.12**
NE+SA	25.03	25.55	86.15	56.25	48.25
E+NSA	24.18	27.50	85.65	56.80	48.53
NE+NSA	20.55	26.38	85.05	56.23	47.05

Table 2: DeepMath-Hard: The "5+2" framework and the sampling algorithm performance across mathematical benchmarks

ablation sturdies are detailed in Appendix O.

4.3 Main Results

4.3.1 Comparison with Other Datasets

In the ablation studies, we have curated datasets of varying sizes from OpenSourceR1-Hard, employing both our "5+2" framework and the sampling algorithm with target data size d set to 1k, 20k, 40k, 60k, respectively. These datasets are labeled as *OpenSourceR1-Hard E+SA (1k)*, *OpenSourceR1-Hard E+SA (1/3)*, *OpenSourceR1-Hard E+SA (2/3)*, *OpenSourceR1-Hard E+SA (1)*, in ascending order of their size.

We benchmark our four sampled datasets derived from OpenSourceR1-Hard against several established open-source datasets (as shown in Appendix L). This evaluation was performed by fine-tuning the Qwen2.5-Math-7B model under the training configurations detailed in subsection 4.1.

Dataset	Size	AIME25	AIME24	MATH500	AMC24	Average
s1K-1.1	1k	10.83	18.08	77.15	37.33	35.85
OS-R1-H E+SA (1k)	1k	16.65	18.35	75.15	39.80	**37.49**
Light-R1-SFT-stage-1	76k	33.05	39.45	88.65	65.53	56.67
OpenR1-Math-94k	94k	30.55	**46.10**	88.95	64.58	57.55
OpenThoughts-114k	114k	29.45	35.28	88.85	62.88	54.12
OS-R1-H E+SA (1/3)	20k	29.45	35.00	87.20	66.30	54.49
OS-R1-H E+SA (2/3)	40k	**38.63**	39.43	**90.55**	67.05	58.92
OS-R1-H E+SA (1)	60k	35.03	44.15	90.25	**68.98**	**59.60**

Table 3: Comparison of Our Datasets with Other Datasets. OS-R1-H stands for OpenSourceR1-Hard

Note that our sampled datasets achieve superior performance compared to all selected open-source datasets, despite being only a fraction of their size. Furthermore, our methods demonstrate efficacy even when applied to considerably smaller datasets. Specifically, the dataset comprising 1,000 instances achieved an accuracy rate of 37.49%, outperforming the 35.85% from s1k-1.1, which is a meticulously curated collection of 1,000 instances through rigorous refinement processes.

4.3.2 Analysis of underthinking phenomenon

In addition to evaluating model accuracy, recent studies have identified an "underthining" phenomenon (Wang et al., 2025) in o1-like LLMs, where the model frequently switches between reasoning trajectories without sufficiently exploring each one. Our "5+2" framework, coupled with the sampling algorithm, is specifically designed to eliminate suboptimal subtrajectories and filter out QA pairs that contain such suboptimal subtrajectories. Therefore, we hypothesize an improvement in model's ability to respond to questions with a reduced number of subtrajectories and a deeper analysis within each subtrajectory. To validate our hypothesis, we analyze the variations in total number of tokens in the reasoning process, the number of subtrajectories and the average number of tokens per subtrajectory before and after fine-tuning with the following datasets:

- *OpenSourceR1-Hard E+SA (2/3)* and *DeepMath-Hard E+SA (2/3)*, datasets where both the "5+2" framework and the sampling algorithm are applied.
- *OpenSourceR1-Hard NE+NSA (2/3)* and *DeepMath-Hard NE+NSA (2/3)*, randomly selected datasets with no additional operation.

In Figure 1 (A), we observe a notable reduction in the total number of tokens involved in the reasoning process within the training datasets when the "5+2" framework and sampling algorithm are employed. For the OpenSourceR1-Hard dataset, the total number of tokens decreased by 15.6% (from 8,586 to 7,247), and for the DeepMath-Hard dataset, a 14.0% reduction (from 8,652 to 7,438) was observed. When evaluating models fine-tuned on these datasets, we noticed a 2.2% decrease (from 5,306 to 5,189) for OpenSourceR1-Hard and a more substantial 9.0% decrease (from 6,994 to 6,363) for DeepMath-Hard, respectively.

In Figure 1 (B), in the training data, we observe an 8.9% increase in the average number of tokens per subtrajectory, rising from 1,389 to 1,513 for OpenSourceR1-Hard when applying *E+SA*. Similarly, DeepMath-Hard shows an 27.6% increase under the same conditions. This phenomenon is also observed in the evaluation results post fine-tuning. Models fine-tuned with OpenSourceR1-Hard exhibits an average increase of 12.5% in the average number of tokens per subtrajectory, rising

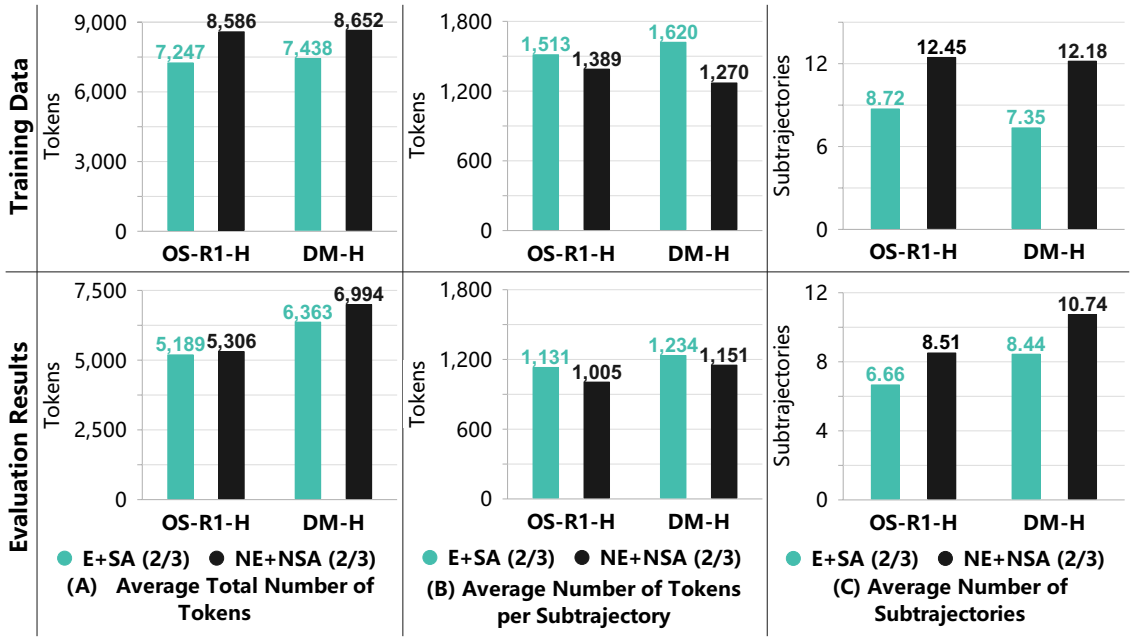

Figure 1: Comparison of Metrics for Thinking Efficacy between Training Data and Evaluation Results, where OS-R1-H stands for OpenSourceR1-Hard and DM-H stands for DeepMath-Hard.

from 1,005 to 1,131 tokens. Similarly, models fine-tuned with DeepMath-Hard shows a 7.2% increase. This implies a deep thinking paradigm during the inference process.

In Figure 1 (C), the application of *E+SA* is notably associated with a significant decrease in the average number of subtrajectories. On the OpenSourceR1-Hard dataset, the average number of subtrajectories decreases from 12.45 to 8.72, a 29.96% reduction. A more pronounced decline: 39.66% is observed on the DeepMath-Hard dataset, with the average dropping from 12.18 to 7.35. A consistent trend is also evident in the evaluation results. When *E+SA* is both applied, a significant reduction in the average number of subtrajectories is observed for models that have been fine-tuned on the OpenSourceR1-Hard and DeepMath-Hard datasets. The average number of subtrajectories decreases by 21.74%, from 8.51 to 6.66, for the OpenSourceR1-Hard dataset. Similarly, for the DeepMath-Hard dataset, the average number of subtrajectories is reduced by 21.41%, decreasing from 10.74 to 8.44.

The empirically findings indicate that the "5+2" framework and the sampling algorithm, or equivalently, models fine-tuned with our datasets effectively mitigate the "underthinking" phenomenon. This is exemplified by a reduction in the number of subtrajectories, coupled with an increase in the number of tokens within each subtrajectory. This outcome signifies a decrease in the frequency of switching approaches and a deeper reasoning within each approach.

4.3.3 Analysis of Suboptimal Subtrajectories in the Evaluation Results

One major aspect of our methods is that the model after fine-tunning with the "5+2" framework and the sampling algorithm is able to generate less number of suboptimal subtrajectories in the evaluation results. Specifically, in Figure 2, model fine-tuned with OpenSourceR1-Hard has a 25.9% (14,234 to 10,554) drop of the number of suboptimal subtrajectories, and with DeepMath-Hard, a 26.4% (18,654 to 13,729) drop of the number of suboptimal subtrajectories with our method applied. See Appendix N for an example.

4.3.4 The Effectiveness of Thinking Budget

To verify the effcetiveness of our method at different thinking budgets, we allocated 1k-16k thinking budgets on the four evaluation benchmarks. The resulting scaling curves are given in Figure 3, *E+SA* demonstrates a significant improvement over *NE+NSA* across the 2k-16k budget range on both OpenSourceR1-Hard and DeepMath-Hard datasets.

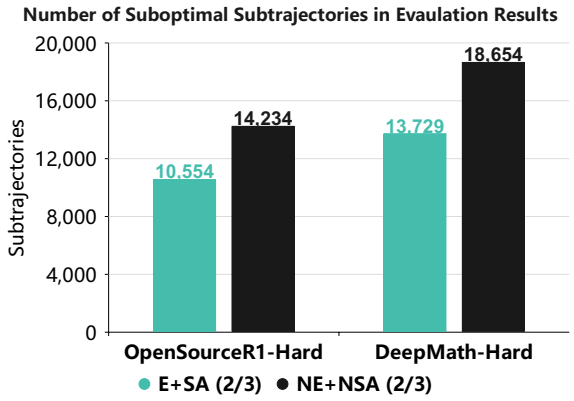

Figure 2: Average Number of Suboptimal Subtrajectories

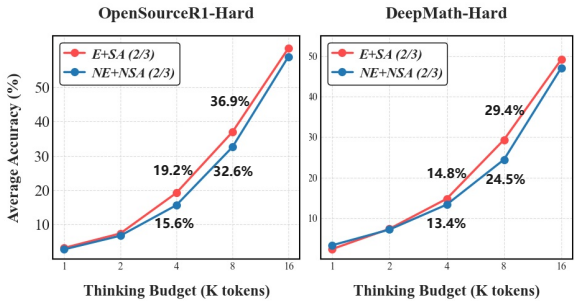

Figure 3: Accuracy of *E+SA* and *NE+NSA* with respect to the thinking budget.

5 Related Work

5.1 Test-Time Scaling

Test-time scaling refers to the practice of enabling LLMs to generate a larger number of tokens during the inference phase, thereby significantly enhancing their problem-solving capabilities. Recent research in this area has primarily focused on two strategies(OpenAI, 2024; Snell et al., 2024): (1) Deploy LLMs to generate multiple reasoning trajectories, from which the optimal path is selected through the application of reward models(Snell et al., 2024; Wu et al., 2024; Brown et al., 2024). Such test-time scaling methods include parallel sampling(Brown et al., 2024; Wang et al., 2022) in which the majority voting mechanism is utilized to select the final answer from multiple generated solutions, and the tree-based search methods(Yao et al., 2023; Zhang et al., 2024; Qi et al., 2024) like Monte-Carlo Tree Search (MCTS). (2) Employ reinforcement learning in the post-training of large LLMs, exemplified by models such as DeepSeek-R1, Qwen-QwQ(Qwen, 2024), DeepSeek-R1(DeepSeek-AI et al., 2025), and Kimi-1.5(Team et al., 2025). These models are capable of exploration, reflection, backtracking, and self-verification, therefore generating significantly longer outputs during inference time.

5.2 Data Selection Policy

It has been empirically demonstrated that high-quality data can enable LLMs to achieve optimal performance with a relatively small number of training samples(Yang et al., 2024b; DeepSeek-AI et al., 2024; Yu et al., 2023). For instance, s1(Muennighoff et al., 2025) demonstrates that a 32B model trained on a dataset of 1,000 samples outperforms OpenAI's o1-preview. Similarly, LIMO(Ye et al., 2025) substantiates the importance of data quality by employing three quality metrics: Optimal Structural Organization, Effective Cognitive Scaffolding, and Rigorous Verification in the selection of training data; a 32B model trained on a dataset of 819 samples, selected through these three criteria, surpasses the performance of o1-preview.

5.3 Thinking Efficacy

Reinforcement learning (RL) enhances model's ability to handle complex reasoning tasks by extending its reasoning process. However, during the problem-solving process, RL forms a unified and specific reasoning paradigm, regardless of the problem's complexity. This paradigm can become highly inefficient if not properly constrained(Ye et al., 2025). In the case of simpler problems, this paradigm may lead to overthinking, as these problems could be resolved with significantly less computational resources(Chen et al., 2024). Conversely, for more complex problems, this paradigm may introduce a significant number of ineffective and counterproductive elements into the reasoning process. Such elements not only compromise the model's accuracy but also diminish its token efficiency. One of such elements is underthinking, where the model switches between strategies too frequently without adequately exploring each one. To mitigate underthinking, (Wang et al., 2025) proposes a decoding strategy that encourages a deeper exploration of each attempted strategy, thereby improving overall accuracy and thinking efficiency. In addition, (Qiao et al., 2025) proposed a ConCISE framework to decrease redundant reasoning steps via confidence strategy in during inference time.

6 Conclusion and Future Works

In this paper, we conducted a comprehensive analysis of the quality of subtrajectories within the

reasoning process of RL-LLMs. This analysis led to the identification of five critical quality issues that negatively impact both the accuracy and thinking efficacy of these models. To address these issues, we introduce a "5+2" framework to: (1) systematically identify suboptimal subtrajectories within the reasoning trajectory based on five human-established criteria; (2) assess the independence of the suboptimal subtrajectories identified in (1) from the subsequent content, ensuring that their elimination does not compromise overall flow and coherence of the reasoning process. Furthermore, we propose a sampling algorithm, built upon the "5+2" framework, to select data that are free from the identified quality issues to the maximum extent. Our experimental findings illustrate that our methods not only improve model accuracy but also enhances thinking efficacy by mitigating the "underthinking" issue, reducing the number of suboptimal subtrajectories, thereby improving the efficacy across different thinking budgets. In the future, we will generalize our method to other disciplines, such as physics and coding. Additionally, we aim to investigate the scalability of our framework by applying it to models of varying sizes, with larger models such as those with 32B parameters or more.

Limitations

1. It is worth noting that while our methods demonstrate significant utility in math domain, where multiple subtrajectories are often presented, domains with fewer number of subtrajectories or those that differ significantly in their reasoning paradigm may necessitate customized frameworks to attain comparable benefits.

2. Our methods primarily emphasize data quality. Beyond quality, the diversity of the QA pairs is also a crucial factor. It is noteworthy that our methods designed to enhance quality may inadvertently lead to imbalances in the diversity distribution of the dataset.

References

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher R'e, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *ArXiv*, abs/2407.21787.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2024. Do not think that much for 2+3=? on the overthinking of o1-like llms. *ArXiv*, abs/2412.21187.

MMA Committees. 2024a. Aime24 problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions/.

MMA Committees. 2024b. Amc24 problems and solutions. https://artofproblemsolving.com/wiki/index.php/AMC_12_Problems_and_Solutions.

MMA Committees. 2025. Aime25 problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions/.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 179 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 179 others. 2024. Deepseek-v3 technical report. *ArXiv*, abs/2412.19437.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning.

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

James M. Joyce. 2011. Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*.

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Fei-Fei Li, Hanna Hajishirzi, Luke S. Zettlemoyer, Percy Liang, Emmanuel J. Candes, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *ArXiv*, abs/2501.19393.

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *ArXiv*, abs/2408.06195.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and Pengfei Liu. 2024. O1 replication journey: A strategic progress report - part 1. *ArXiv*, abs/2410.18982.

Qwen. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Feng Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *ArXiv*, abs/2501.12599.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *ArXiv*, abs/2501.18585.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. 2025. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *ArXiv*, abs/2503.10460.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report.

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. 2024a. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *ArXiv*, abs/2409.12122.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, and 25 others. 2024b. Qwen2.5 technical report. *ArXiv*, abs/2412.15115.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

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

Long Long Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zheng Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *ArXiv*, abs/2309.12284.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *ArXiv*, abs/2406.07394.

A Example of DeepSeek-R1-distilled QA Pairs

Example of DeepSeek-R1-distilled QA Pairs

Question: Let the circles k_1 and k_2 intersect at two distinct points A and B, and let t be a common tangent of k_1 and k_2, that touches k_1 and k_2 at M and N, respectively. If $t \perp AM$ and $MN = 2AM$, evaluate $\angle NMB$.

Think:

<think>

Alright, let's try to tackle this problem step by step ...

Alternatively, since O_1M and O_2N are both perpendicular to t ...

Alternatively, maybe the problem is from a math competition like the AIME, where n is given, ...

Alternatively, since AM is perpendicular to t, and MN is along t, then ...

...

</think>

Final Answer: Given two intersecting circles k_1 and k_2 with common points A and B, ... The measure of $\angle NMB$ is $\boxed{45^\circ}$.

B Data Curation

B.1 Filtering Processes

B.1.1 Basic Quality Filtering

We design a rule-based filtering task, aiming to remove any entries that meet certain predefined criteria, including: (1) image-dependent questions: questions that require images, in forms of hyperlinks or visual references, to be answered; (2) truncated solutions: solutions that are cut off due to the length of the model output exceeding the predefined maximum token limit; (3) inconsistent language use: entries with mixed or incoherent language, such as abrupt shifts between English and Chinese.

B.1.2 Difficulty Filtering

We implement a two-stage difficulty filtering process, similar to the ones employed in s1 (Muennighoff et al., 2025) and LIMO (Ye et al., 2025). The primary objective of this process is to only retain those entries that contain questions requiring highly complex and intricate reasoning solutions, thereby exceeding the capabilities of the current base models. For each entry (question, R1 solution), we deploy two models: Qwen2.5-Math-7B-Instruct (Yang et al., 2024a) and R1-Distill-Qwen-7B (DeepSeek-AI et al., 2025), to independently generate answers twice. Following the generation, we deploy a third 7B model, specifically fine-tuned for the purpose of assessing the correctness of generated answers in comparison to the ground truth. In this scenario, the 7B model evaluates the generated answers against the final answer extracted from the R1 solution. We exclude any entries where either model provides a correct answer at least once, and any entries in which there is no clearly marked final answer, i.e., boxed{}, in the R1 solution.

B.2 Curation of OpenSourceR1-Hard

We collect 5 open-source R1-distilled datasets from Hugging Face, totaling 210k samples after deduplication. The basic information of the collected datasets is listed as follows:

Dataset	Dataset Size
OpenThoughts-114k[1]	114k
OpenR1-Math-94k[2]	94k
s1K-1.1[3]	1k
Light-R1-SFT-stage-1[4]	76k
Light-R1-SFT-stage-2[5]	3k

Table 4: Basic Information of Collected Datasets

After applying both basic quality filtering and difficulty filtering, we curated a dataset of 59,759 entries, which we will refer to as OpenSourceR1-Hard in later discussions. We remark that our hypotheses and methodologies presented in later sections are both formulated and validated using OpenSourceR1-Hard, thereby making it an in-distribution dataset.

B.3 Curation of DeepMath-Hard

During the preparation of this paper, we came across a recently released dataset called DeepMath (He et al., 2025), a 103K R1-distilled dataset. We

[1] https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k

[2] https://huggingface.co/datasets/llamafactory/OpenR1-Math-94k

[3] https://huggingface.co/datasets/simplescaling/s1K-1.1

[4] https://huggingface.co/datasets/qihoo360/Light-R1-SFTData

[5] https://huggingface.co/datasets/qihoo360/Light-R1-SFTData

apply the same filtering processes to DeepMath, including both the basic quality filtering and difficulty filtering, albeit with slightly adjusted sampling parameters. This reduces the original DeepMath dataset to a more compact version, referred to as DeepMath-Hard, which consists of only 12,719 entries. The rationale for treating OpenSourceR1-Hard and DeepMath-Hard as separate datasets, rather than concatenating them, stems from the fact that our hypotheses and methodologies are both formulated and validated using OpenSourceR1-Hard. To evaluate the generalization of our approach, we deliberately isolated the DeepMath-Hard dataset, which will function as an out-of-distribution test set.

C Examples of 5 types of subtrajetories

1. *The subtrajectory proposes a method without attempting it.*

   Example 1

   Alternatively, this is similar to a three-dimensional matching problem, which is NP-hard, but maybe in this specific case, with the constraints on the digit sums, it can be solved more easily.

   Example 1 mentions a three-dimensional matching problem; however, it falls short by not stating a precise definition of the problem, its relevance to the current context, and any attempt to address and resolve the problem through this approach.

2. *The subtrajectory attempts to solve the problem in an ineffective manner.*

   Example 2

   Alternatively, . . . Let's start testing numbers step by step, starting from the smallest natural numbers, checking if they meet the criteria. Starting with n=1: . . . n=2: . . . n=119: . . . n=120: . . . This is getting tedious. . .

   Example 2 evaluates numbers from 1 to 120 in a mechanical manner, without an attempt to identify any underlying patterns that could have simplified or advanced the process, thereby leading to a lengthy and ineffective argument.

3. *The subtrajectory has logical discontinuities.*

   Example 3

   Alternatively, . . . This is getting very complicated. Given that we already found a critical point at $a = 1, b = 1$, and that when we check other points, A is higher, perhaps we can conjecture that the minimal value is $\frac{2}{\sqrt{5}}$. To confirm, let's check the second derivative or the behavior around $t = 1$, but since it's time-consuming and given the complexity, I think the minimal value is indeed $\frac{2}{\sqrt{5}}$.

   Example 3 contains a logical gap, as it circumvents rigorous computation and instead relies on speculative assumptions regarding the outcome.

4. *The subtrajectory transitions to the next one without reaching any conclusions.*

   Example 4

   Alternatively,. . . Let me think about how to approach this. Since each number is a three-digit number without any zeros, each digit is from 1 to 9, and their sum is 9. So first, maybe I should figure out all possible three-digit numbers that satisfy conditions 1 and 2, and then see how many of them can be selected such that conditions 3, 4, and 5 are also satisfied.
   Alternatively, since the digits in each place. . .

   Example 4 first proposes Approach A; however, it abruptly shifts to Approach B without concluding Approach A, assessing its efficacy, elaborating on the subsequent steps and associated challenges, or explaining the rationale for abandoning Approach A.

5. *The subtrajectory contains redundant self-verification(s).*

Example 5

Alternatively,... Case 3: $p = 2$, $q = 7$. Compute numerator: $2^7 - 7^2 = 128 - 49 = 79$...
Alternatively,... Wait, perhaps check $p = 2$ and $q = 7$ again. Wait, $p = 2$, $q = 7$ gives $128 - 49 = 79$. 79 divided by 9 is $8.777\ldots$, which is not integer...
Alternatively,... Wait, maybe $p = 2$, $q = 7$, numerator= 79. 79 is a prime. 79 divided by 9, which is not divisible. So, no.

Example 5 redundantly verifies the same case twice using the same method.

D Example of Non-eliminable Suboptimal Subtrajectory

Example of Non-eliminable Suboptimal Subtrajectory

Alternatively, ... using the formula:
Area $= \frac{1}{2}|(x_A - x_P)(y_B - y_P) - (x_B - x_P)(y_A - y_P)|$.
But perhaps a better approach is to find the coordinates of A and B in terms of h, then compute vectors PA and PB, and compute the cross product area.
Alternatively, since we know the equation of line AB is $y = hx - h + 2$, and point P is $(h, h-2)$. Then, the area of triangle PAB can be calculated as $\frac{1}{2} \cdot$ base $\cdot$ height where the base is the distance between A and B, and the height is the distance from P to the line AB ...
Alternatively, ...
Area $= \frac{1}{2}|(x_A - x_P)(y_B - y_P) - (x_B - x_P)(y_A - y_P)|$.
Plugging in the coordinates: ...

E Equal Weights

Here, each subtrajectory is considered to hold equal importance. Specifically, we aggregate the scores of all subtrajectories according to the following formula:

$$\begin{aligned} &\mathit{QualityScore}(\text{thinking}) := \\ &\sum_{i=1}^{n} \frac{1}{n} \left(\mathit{Score}(\text{subtrajectory})\right), \end{aligned} \tag{3}$$

where n is the number of subtrajectories within the thinking process.

F Distribution of Number of Subtrajectories after Sampling on Quality Score

Figure 4 presents the distribution of number of subtrajectories between the entire dataset and top 1/3 of data by quality scores, and we can find the distribution is clearly shifting towards a direction with fewer numbers of subtrajectories after quality filtering.

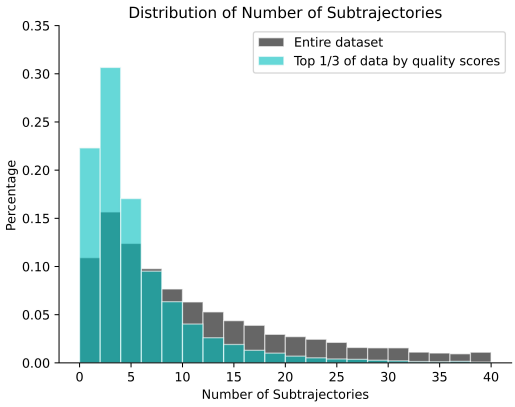

Figure 4: Number of Subtrajectories within QA Pairs Selected by Quality Scores

G Sampling Algorithm

1. Define the following parameters:

   $\mathit{Entire} \leftarrow$ entire dataset,
   $d \leftarrow$ target size for the sampled, dataset
   $I \leftarrow$ set of all possible numbers of subtrajectories within the thinking process in a QA pair in Entire

2. For each QA pair, calculate its quality score, denoted as $\mathit{QualityScore}$(QA pair).

3. Sort Entire by $\mathit{QualityScore}(\cdot)$ descendingly. Select the top d QA pairs to form the subset

   $\mathit{Pseudo\_Sampled\_init}$.

4. For $i \in I$, calculate the percentage change in the frequency of QA pairs whose the thinking process involves precisely i subtrajectories, relative to the entire dataset:

$$\Delta_i = \frac{F_E(i) - F_{PSinit}(i)}{F_E(i)}, \quad i \in I, \quad (4)$$

where $F_E(\cdot)$ and $F_{PSinit}(\cdot)$ denote the frequencies of QA pairs whose thinking process contains exactly $\cdot$ subtrajectories in $Entire$ and $Pseudo\_Sampled\_init$, respectively.

5. For each QA pair, get the number of subtrajectories within its thinking process, denoted by n. For $0 \leq j \leq 40$, compute:

$$\begin{aligned} &SamplingScore_j(\text{QA pair}) \\ &= \alpha_j QualityScore(\text{QA pair}) \\ &+ (1-\alpha_j)\frac{\Delta_n - \min(\{\Delta_i\}_{i\in I})}{\max(\{\Delta_i\}_{i\in I}) - \min(\{\Delta_i\}_{i\in I})}, \quad (5) \\ &\alpha_j = \frac{3}{5} + \frac{j}{100}. \end{aligned}$$

We remark that the weight α_j ranges from 0.6 to 1.0, rather than from 0.0 to 1.0, is to place a larger emphasis on the $QualityScore$ and to prevent the minimization of KL divergence from dominating the data sampling process.

6. For $0 \leq j \leq 40$, sort $Entire$ by $SampleScore_j(\cdot)$ descendingly. Select the top d QA pairs to form the subset $Pseudo\_Sample\_j$.

7. The sampled dataset

$$Sampled := \underset{Pseudo\_Sample\_j}{\arg\min} D_{KL}(X_E || X_{PSj}), \quad (6)$$

where $D_{KL}(\cdot)$ is the Kullback-Leibler (KL) divergence, X_E and X_{PSj} are the distribution of number of subtrajectories within the thinking process in $Entire$ and $Pseudo\_Sample\_j$, respectively.

H Training configurations

Each training process employs full-parameter fine-tuning and consumes 576 Ascend 910B4 NPU hours separately. The hyperparameters for training are configured as follows: training steps = 24,000, batch size = 5, max sequence length = 16,384 tokens, trainings are performed in bfloat16 precision, learning rate is initially set to 2e-5, linearly warms up for 1% of the total training steps, and decays to 2e-9 following a cosine schedule, optimization is performed using the AdamW algorithm with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$.

I Benchmarks

- **American Invitational Mathematics Examination (AIME24 (Committees, 2024a), AIME25 (Committees, 2025))**: a mathematical competition consisting of two examinations: AIME I and AIME II, each containing 15 questions. The AIME examination covers a broad range of mathematical topics, including arithmetic, algebra, combinatorics, etc.

- **MATH500** (Hendrycks et al., 2021): a collection of high school-level competition problems spanning seven subjects (Algebra, Number Theory, Geometry, etc.) and five difficulty levels, ranging from the easiest problems in the AMC 8 to the most challenging problems in the AIME.

- **American Mathematics Competitions (AMC24 (Committees, 2024b))**: the initial examination administered by the Mathematical Association of America before qualifying for the AIME. In 2024, the AMC12 consisted of 50 questions, from which we excluded those involving graphs, leaving us with 44 questions for evaluation.

J Evaluation Methods

When evaluating the performance of SFTed models, we employ the pass@1 metric across all evaluation benchmark under a Zero-shot Chain-of-Thought configuration. Furthermore, we take four recent checkpoints, corresponding to training stages of 18k, 20k, 22k, 24k steps, respectively. Each checkpoint is evaluated three times for AIME24, AIME25, and AMC24, and once for MATH500; following this, we compute and report the average accuracy for each benchmark, based on the 12 ($3 \cdot 4$) evaluations for AIME24, AIME25, and AMC24, and 4 ($1 \cdot 4$) evaluations for MATH500. Throughout the evaluation, we set the temperature at 0.7 and impose a maximum output length constraint of 16,384 tokens.

We will explain the rationale behind our decision to report the average accuracy of multiple checkpoints, rather than make the standard practice of selecting the single checkpoint with the lowest validation loss. Our benchmarks, including AIME24, AIME25, and AMC24, consist of a limited number of questions. Consequently, a fortunate checkpoint that achieves two additional correct answers could result in a fluctation of up to 6.6%, significantly

skewing the evaluation results. To mitigate this variability and enhance the stability of our performance metrics, we have opted for the strategy of averaging the accuracy of recent checkpoints. This strategy aims to provide a more reliable and consistent assessment of our methods.

K Ablation Settings

1. **Elimination with Sampling Algorithm (*E+SA*)**
   We incorporate all proposed modules within our methods. Specifically, we use the "5+2" framework to identify and eliminate suboptimal subtrajectories and assign token-count-based weights to evaluate the thinking process. Subsequently, we employ the sampling algorithm tailored to the appropriate target size to select the data.

2. **No Elimination with Sampling Algorithm (*NE+SA*)**
   This configuration preserves the identical set of questions as those utilized in the *E+SA* and maintains the original solutions, thereby facilitating a rigorous comparison of the impact introduced by the "5+2" framework.

3. **Elimination without Sampling Algorithm (*E+NSA*)**
   The questions are randomly selected from the entire dataset. Furthermore, we employ the "5+2" framework to identify and eliminate suboptimal subtrajectories.

4. **No Elimination without Sampling Algorithm (*NE+NSA*)**
   This configuration retains the identical set of questions as those utilized in the *E+NSA*. However, we employ the original solutions to facilitate a comparative analysis with *E+NSA*, thereby enabling an assessment of the efficacy of the "5+2" framework.

L Details of the sampled dataset

- **s1K-1.1**: a dataset comprising 1k diverse, high-quality and challenging QA pairs with answers generated by DeepSeek-R1.

- **Light-R1-SFT-stage-1**: a dataset consisting of 76k samples, sourced from publicly available mathematics datasets.

- **OpenR1-Math-94k**: a dataset consisting of 94k problems, extracted from the larger OpenR1-Math-220k dataset. Despite its smaller size, this dataset exhibits superior performance compared to the entire 220k dataset. Each question originates from NuminaMath1.5, with corresponding answers generated by DeepSeek-R1.

- **OpenThoughts-114k**: a comprehensive synthetic reasoning dataset consisting of 114k samples across mathematics, science, coding, and puzzles.

M Demonstration of Varied Weights Based on Token Counts

The Figure 5 demonstrates the varied weights based on token counts.

N Examples of Inference Outputs

Figure 6 illustrates a comparative analysis between the inference outputs of AIME25 utilizing the configurations *E+SA (2/3)* and *NE+NSA (2/3)*.

O Ablation Studies

O.1 Varied Weights vs Equal Weights

In this study, we introduced two approaches for scoring a thinking process after eliminating suboptimal subtrajectories: 1) equal weights scoring process (see details in Appendix E), and 2) varied weights scoring process (see details in section 3.3.2). Our experiments are conducted using the OpenSourceR1-Hard and DeepMath-Hard datasets. The experimental analyses are conducted on two-thirds of each respective dataset, and for each sample fraction, the following two configurations are considered:

1. **Varied Weights**
   This configuration mirrors the *E+SA* setup. Specifically, we employ the "5+2" framework to identify and eliminate suboptimal subtrajectories, and assign weights based on the token counts of each subtrajectory during the scoring process. Subsequently, we utilize the sampling algorithm to select two-thirds of the dataset.

2. **Equal Weights**
   We employ the "5+2" framework to identify and eliminate suboptimal subtrajectories, assigning equal weights to each subtrajectory

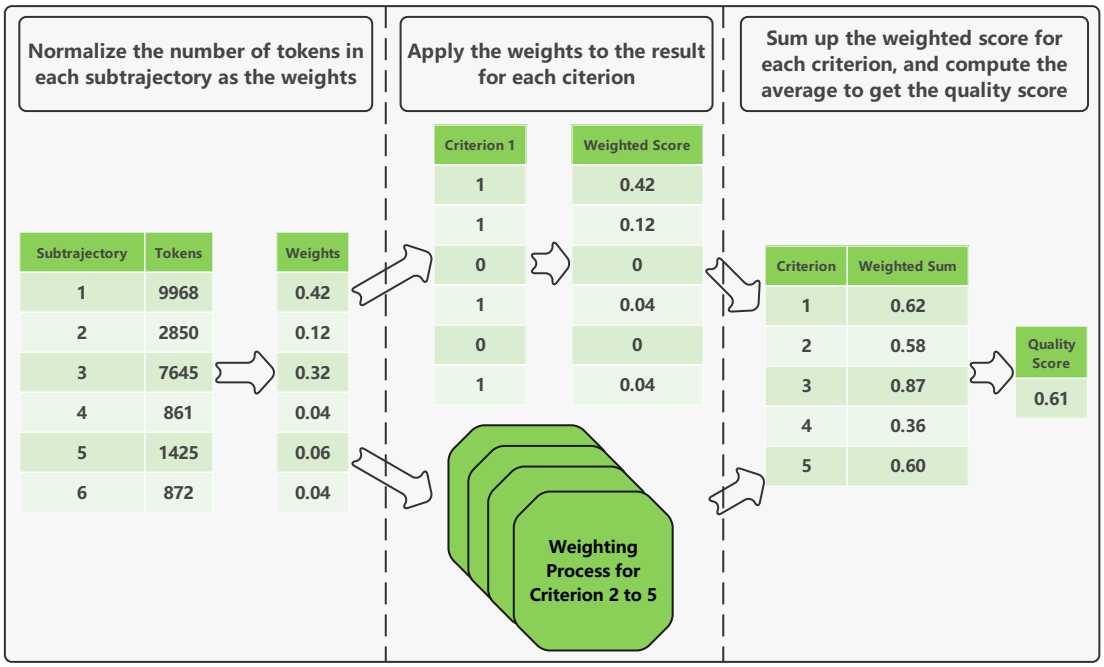

Figure 5: Demonstration of Varied Weights Based on Token Counts

during the scoring process, prior to applying the sampling algorithm to select two-thirds of the dataset.

As presented in Table 5, the varied weights scoring method, utilizing the OpenSourceR1-Hard dataset, achieves a performance of 58.92%, outperforming the 56.74% accuracy obtained through the equal weights scoring method. Similarly, when employing the DeepMath-Hard dataset, the accuracy improves from 47.94% (equal weights) to 49.12% (varied weights). These findings suggest that the token counts of subtrajectories are crucial for assessing the quality of solutions. Specifically, longer suboptimal subtrajectories should be subjected to greater penalties compared to shorter ones in the evaluation of overall performance.

Methods	AIME25	AIME24	MATH500	AMC24	Average
OpenSourceR1-Hard					
Varied Weights	38.63	39.43	90.55	67.05	**58.92**
Equal Weights	29.73	41.40	89.35	66.48	56.74
DeepMath-Hard					
Varied Weights	27.23	27.23	85.20	56.80	**49.12**
Equal Weights	24.98	23.88	85.10	57.78	47.94

Table 5: Equal weight scoring process and token-count based scoring process comparison in mathematical domain: performance across various benchmarks

O.2 Impact of Sampling Algorithm

The experiments employ the OpenSourceR1-Hard and DeepMath-Hard datasets. For both datasets, the experiments are conducted on a two third of the dataset. Two distinct experimental conditions are established for each dataset to facilitate comprehensive evaluation: with sampling algorithm and without sampling algorithm.

1. **With Sampling Algorithm**
   This configuration is identical to the *E+SA* setup. Following the identification and elimination of suboptimal subtrajectories according to the "5+2" framework, a token-count based weighting scoring process is applied to the thinking process. Subsequently, the sampling algorithm is implemented to select two-thirds of the dataset.

2. **Without Sampling Algorithm**
   This configuration undergoes the identification and elimination of suboptimal subtrajectories based on the "5+2" framework, followed by the application of a token-count based weighting scoring process to the thinking process.

The findings are elaborated in Table 6. Specifically, in the OpenSourceR1-Hard dataset, the performance metric with the sampling algorithm (58.92%) surpasses that without the sampling algorithm (58.60%). A similar trend is observed in the DeepMath-Hard dataset, where the result obtained with the sampling algorithm (49.12%) is superior to the result without it (48.57%).

These observations imply that the distribution of the number of subtrajectories within the dataset can influence the SFT process. By penalizing the significant variation in the frequency of number of subtrajectories within the sampled dataset's thinking process relative to the entire dataset, an additional enhancement in the model's performance is achieved.

Methods	AIME25	AIME24	MATH500	AMC24	Average
OpenSourceR1-Hard					
w/ Sampling Algorithm	38.63	39.43	90.55	67.05	**58.92**
w/o Sampling Algorithm	33.33	43.35	89.50	68.20	58.60
DeepMath-Hard					
w/ Sampling Algorithm	27.23	27.23	85.20	56.80	**49.12**
w/o Sampling Algorithm	23.88	28.05	84.00	58.35	48.57

Table 6: Sampling algorithm comparison in mathematical domain: performance across various benchmarks

Comparative Analysis of Evaluation Results with and without Implementation of Our Methods

Question: In $\triangle ABC$ points D and E lie on $\overline{AB}$ so that $AD < AE < AB$, while points F and G lie on $\overline{AC}$ so that $AF < AG < AC$. Suppose $AD = 4$, $DE = 16$, $EB = 8$, $AF = 13$, $FG = 52$, and $GC = 26$. Let M be the reflection of D through F, and let N be the reflection of G through E. The area of quadrilateral $DEGF$ is 288. Find the area of heptagon $AFNBCEM$.

E+SA (2/3)

Okay, let's try to tackle this problem. It seems a bit involved with reflections and areas in a triangle. I need to find the area of a heptagon AFNBCEM given some information about the original triangle ABC and various points. Let me start by understanding the problem step by step.
[ommited for brevity]
Let me assign coordinates to the triangle ABC. ...However, in reality, **triangle ABC isn't necessarily right-angled. Hmm, this complicates things. Maybe we need to use ratios instead or find coordinates based on the given lengths.**
Alternatively, maybe compute areas using vectors or coordinate geometry.
[ommited for brevity]
Now, we need to compute the area of heptagon AFNBCEM. Let's list all the vertices in order: $A(0,0)$, $F(\sqrt{}(6517)/7, 6)$, $N(40 - 5 * \sqrt{}(6517)/7, -30)$, $B(28,0)$, $C(\sqrt{}(6517), 42)$, $E(20,0)$, $M(2 * \sqrt{}(6517)/7 - 4, 12)$, **back to** $A(0,0)$.
This seems complex due to the square roots. However, maybe there is symmetry or cancellation that can simplify the calculation.
Alternatively, use the shoelace formula with symbolic coordinates. Let's attempt that.
[ommited for brevity]
Now, applying the shoelace formula:
Sum1 = $(x_A y_F + x_F y_N + x_N y_B + x_B y_C + x_C y_E + x_E y_M + x_M y_A)$
Sum2 = $(y_A x_F + y_F x_N + y_N x_B + y_B x_C + y_C x_E + y_E x_M + y_M x_A)$
[ommited for brevity]
So the area of the heptagon AFNBCEM is 588.
Wait, that's surprisingly straightforward. Let me **verify** the steps again.
[ommited for brevity]
we arrived at the heptagon area being 588. Let me check if this is plausible.
Alternatively, maybe there's a relationship between DEGF and the heptagon.
[ommited for brevity]
Therefore, despite the complex coordinates, the area simplifies neatly due to cancellation in the shoelace formula. Therefore, the area of heptagon AFNBCEM is 588.
Final Answer $\boxed{588}$

NE+NSA (2/3)

Okay, let's try to tackle this geometry problem. It seems a bit complex at first glance, but I'll take it step by step.
[ommited for brevity]
Wait, but maybe we can assume that triangle ABC is right-angled? Wait, no, there's no information given about angles, so we can't assume that. ***Alternatively*, perhaps assign coordinates based on ratios?** [Violate Criterion 1,2,3,4]
Alternatively, since we know the area of quadrilateral DEGF is 288. If we can model the coordinates accordingly, then compute the area of the heptagon.
[ommited for brevity]
Similarly, coordinates of G would be $(65/91 * c_x, 65/91 * c_y)$ since AG = 65.
Alternatively, since AC is length 91, **we can use vectors or coordinates scaled accordingly.** [Violate Criterion 1,2,3,4]
But perhaps if I assign coordinates to C such that calculations are manageable. Let me suppose that AC is along some line, but maybe assign coordinates to make computation easier.
Alternatively, set coordinate system such that AC is vertical? Wait, but unless there is a specific angle, that might complicate things. ***Alternatively*, use barycentric coordinates? Not sure.** [Violate Criterion 1,2,3,4]
Alternatively, since we know the area of quadrilateral DEGF is 288. Let's think about quadrilateral DEGF. Points D(4,0), E(20,0), G, F.
[ommited for brevity]
***Alternatively*, perhaps use mass point geometry? Not sure.** [Violate Criterion 1,2,3,4]
[ommited for brevity]
***Alternatively*, break the heptagon into simpler parts whose areas can be calculated and summed.** [Violate Criterion 1,2,3,4]
Alternatively, use the shoelace formula with the coordinates.
[ommited for brevity]
Therefore, the area of the heptagon AFNBCEM is 588.
[ommited for brevity]
Given that all steps check out and the mathematical derivation leads to 588, I think the answer is 588.
Final Answer $\boxed{588}$

Figure 6: Examples of Inference Outputs