An Empirical Study of Activating Slow-thinking Capability of Large Language Models

Anonymous ACL submission

Abstract

Recently, slow-thinking reasoning systems, such as o1, have demonstrated remarkable capabilities in solving complex reasoning tasks. These systems typically engage in an extended thinking process before responding to a query, allowing them to generate more thorough, accurate, and well-reasoned solutions. These systems are primarily developed and maintained by industry, with their core techniques not publicly disclosed. In response, an increasing number of studies from the research community aim to explore the technical foundations underlying these powerful reasoning systems. To reveal the LLM reasoning mechanisms, this paper presents an empirical study on implementing o1-like reasoning systems, focusing on two key questions: (1) *How can LLM learn this reasoning approach* and (2) *How can LLM further improve its reasoning ability without additional demonstration data*. Concretely, we first design an "imitate, explore, and self-improve" framework as our primary technical approach to training the reasoning model. Then, we conduct the experiment to analyze the influence of different selection strategies of training instance and backbone model, and explore the effect of the self-improving process. Following the findings in our experiments, we finally train a powerful LLM, which can perform complex reasoning processes, demonstrating superiority in solving challenging reasoning problems. Our models and data will be publicly released.

1 Introduction

Recently, slow-thinking reasoning systems, exemplified by OpenAI's o1 model [1], have significantly enhanced the capabilities of large language models (LLMs) (Zhao et al., 2023) in tackling challenging tasks (Daniel, 2017; OpenAI, 2024b). Unlike previous reasoning approaches (Wei et al., 2022a; Shao et al., 2024), these systems employ test-time scaling, allowing more time for contemplation before responding to a query. This thinking process is also reflected as a text generation process that produces long internal chains of reasoning steps, referred to as *thoughts*, to discover suitable solutions. By examining the generated thought data, we can observe various complex reasoning behaviors exhibited by LLMs, such as planning, divide-and-conquer, self-refinement, and backtracking. Initially, it may seem surprising that LLMs can manage such complex reasoning processes, even though we know that specific training or inference strategies are employed to support this capability.

To uncover the underlying mechanisms, the research community has been actively exploring slow-thinking reasoning systems via leveraging external signals to guide the reasoning process and further improve the LLM (Jiang et al., 2024b; Zhang et al., 2024b; Zhao et al., 2024; Wang et al., 2024a). However, these implemented systems might not be the correct path toward developing o1-like systems, because of the following three major challenges. First, the domain-specific reward model we trained does not generalize well across different domains. Second, performing tree search during the inference stage was very time-consuming, making it impractical for real-world applications. Third, although test-time scaling works, we still cannot achieve train-time scaling to improve model performance. These considerations have led us to reconsider our technical approach to creating o1-like reasoning systems.

Given the released API or checkpoints for o1-like systems from DeepSeek and Qwen (Team, 2024a,c), we closely examine the actual thought processes rather than the summarized versions in o1. LLMs can perform complex thought processes without guidance from external signals (Qin et al., 2024; Huang et al., 2024). Given this phenomenon, we consider whether leveraging only a small amount of annotated long chain-of-thought data can significantly enhance the performance

[1] https://openai.com/o1/

of LLMs on complex reasoning tasks. Based on these considerations, in this work, we proposed a comprehensive empirical study about the training paradigm of slow-thinking systems, exploring two key questions: (1) How can LLM learn this reasoning approach and (2) How can LLM further improve its reasoning ability without additional demonstration data.

Specifically, we first propose a conceptual framework comprising an "*imitate, explore, and self-improve*" process for conducting our empirical study. In this framework, LLMs imitate the reasoning approach (*i.e.,* long chain-of-thought reasoning) from a small number of labeled instances, and then explore the possible solution to the given problems without the demonstration and learning from the successful trajectories. Next, based on this framework, we assess the influence of different selection strategies on the demonstration data and the backbone model, and analyze the effect of different self-improving methods. According to the empirical experiments, we observe that the reasoning ability of LLM can generalize from the source domain (*i.e.,* mathematics) to other domains (*i.e.,* physical, chemistry, and biology), and we also find that data selection strategy is really important in the training process, which greatly influences the final performance of LLM. Moreover, through training on these demonstration instances, LLM possesses the ability to enhance itself, even in different domains. Finally, following the findings in our empirical study, we successfully train an LLM that shows remarkable performance on complex reasoning tasks. In summary, our contributions are as follows:

(1)We conduct the empirical study about the slow-thinking LLM training process, and we observe that only with a small amount of carefully selected demonstration instances, LLM can possess the amazing ability to solve challenging reasoning problems and domain generalization.

(2)We investigate the effectiveness of LLM self-improving, and find that once the LLM controls the reasoning approach, it can try to generate the responses for the unlabeled instances and further improve itself via these exploration trajectories.

(3)Based on the findings in our empirical study, we successfully train a slow-thinking LLM, achieving very promising results on college-level and competition-level problems, including MATH-OAI (Lightman et al., 2024), AIME [2], and GPQA (Rein et al., 2023)

2 Related Work

System 1 and System 2. Previous study (Kahneman, 2011) has pointed out the different reasoning methods, *i.e.,* system 1 and system 2. In the context of LLMs, the former refers to the model directly generating the final answer (Yu et al., 2023; Radford et al., 2019), while the latter requires the model first to generate the thinking process and then generate the final answer (Weston and Sukhbaatar, 2023; Deng et al., 2023). According to the generated intermedia reasoning steps, system 2 outperforms system 1 in various tasks (Wei et al., 2022b; Kojima et al., 2022). However, it requires more resources than system 1 during the reasoning process, and how to reduce the inference time is also a critical research problem (Yu et al., 2024; Zhang et al., 2024c). In this work, we focus on building a slow-thinking LLM, *i.e.,* a system 2 model, which can better solve complex reasoning problems.

Chain-of-Thought Reasoning. Chain-of-Thought (CoT) prompting strategy utilizes exemplars or instructions to guide LLMs to generate the reasoning steps in natural language format before obtaining the final answer (Wei et al., 2022b; Kojima et al., 2022). Inspired by CoT, a surge of work induces LLMs to generate the intermedia reasoning steps in the different formats, *e.g.,* code (Gao et al., 2022), tree (Yao et al., 2023), or graph (Besta et al., 2024). Moreover, external tools or guidance can also be leveraged to further enhance the reasoning ability of LLMs (Gou et al., 2023; Schick et al., 2023; Chen et al., 2023; Jiang et al., 2024a). In this work, we train LLM to perform long-form thought and then generate the solution based on the thought without integrating external guidance, which is a special form of CoT.

Self-evolution on LLMs. Given the limitation of supervised data, the existing study has leveraged the LLM self-generated data to train the model itself (Wang et al., 2023; Huang et al., 2023; Tao et al., 2024). To enhance the effectiveness of self-evolution, previous work adopts external signals to improve the quality of training data (Zhang et al., 2024a; Lu et al., 2024), or design fine-grained su-

[2]https://huggingface.co/datasets/AI-MO/aimo-validation-amc

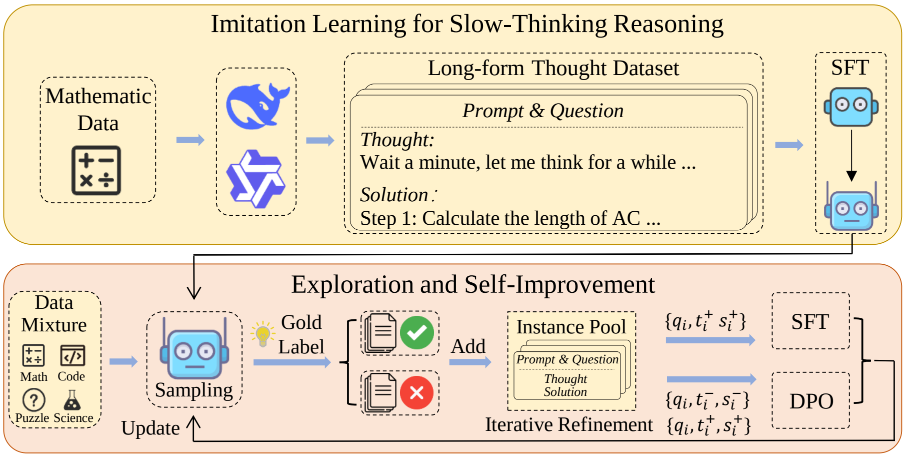

Figure 1: An illustrative overview of our training pipeline.

pervision to make LLMs better learn the corresponding knowledge (Wang et al., 2024b; Chen et al., 2024). In this work, our major concern is how to utilize the existing supervised instances to construct the training dataset, which effectively and efficiently makes LLM possess the ability to perform long-form reasoning. This ability can be utilized in the solve-evolution process.

3 Method

In this section, we provide a detailed description of how to implement o1-like reasoning systems[3].

3.1 Overview

In this work, we propose a two-phase training approach—*imitate, explore, and self-improve*—to develop reasoning systems similar to o1. After training, the inference phase is also completed by a single-pass text generation process, akin to prior prompt-based methods, with the key distinction that the generated response includes both the reasoning process and the solution. We show the overview of our method in Figure 1. Next, we detail each phase below.

• *Imitate*: The core idea is that both the internal thought process and the final solution should be generated in a single response. We argue that a well-established model, even with a small amount of long-form thought data, can easily adhere to o1-like output formats, which is fundamentally about following a prescribed format. The key rationale is that, although the entire thought process may be complex, LLMs can effectively handle individual steps (*e.g.,* planning, verification, and refinement). By using format-following, we can guide LLMs to seamlessly manage and connect these steps. If this hypothesis proves true, two major benefits can be realized: (1) large amounts of data are unnecessary for format-following, and (2) the approach can be easily generalized to various domains.

• *Explore and Self-Improve*: While imitation enables LLMs to generate o1-like outputs, it may not fully encourage the model to master or improve its ability to use long-form thought to tackle complex tasks. To address this, we further incorporate *exploration* to allow the model to find the correct *trajectory* (*i.e.,* the entire response consisting of thought and solution) by sampling multiple candidate solutions to refine the training data. And we evaluate the correctness of these attempted solutions by comparing them with the golden labels. Then, we can further enhance the model's reasoning capabilities by utilizing progressively improved trajectory data to perform supervised fine-tuning or direct preference optimization. We hypothesize that providing high-quality demonstrations—particularly those the model cannot easily generate—will effectively strengthen its reasoning abilities.

3.2 Slow-Thinking Reasoning Imitation

As discussed in Section 1, we propose using imitation learning to enable the LLM to engage in slow-thinking reasoning—producing an extended process of thought (referred to as *long-form thought*[4]) before responding to a query.

[3]Because the exact development of OpenAI's o1 systems is not publicly known, in this paper, "o1-like" refers to the reasoning systems that first conduct extensive reasoning process before producing the final solution.

[4]We prefer not to use "chain-of-thought" since thoughts can be presented flexibly, embodying different reasoning structures.

3.2.1 Long-form Thought Data Construction

To guide the LLM in a slow-thinking mode, we first need to construct a collection of high-quality demonstration data that exhibits this behavior.

Given the simplicity and budget constraints, we collect long-form thought data from existing o1-like reasoning systems, such as *DeepSeek-R1-Lite-Preview* (Team, 2024a) (abbreviated as *R1*) and *QwQ-32B-preview* (Team, 2024c) (abbreviated as *QwQ*), which provide the detailed long-thought process. Our goal is to develop more generalized LLMs capable of reasoning across different domains. To achieve this, we begin by collecting demonstration instances from the mathematic domain, as we hypothesize that the ability to perform long-form reasoning can transfer easily across them. After collecting the labeled data, we perform further pre-processing to ensure data quality, including deduplication and filtering. We show the details of the long-form thought dataset construction process, including data collection, format unification, and data pose-process in Appendix A.

3.2.2 Long-form Thought Instruction Tuning

After collecting instruction data for long-form reasoning, we fine-tune the model to replicate the behavior of the slow-thinking mode. Specifically, we first determine the data ratio for each domain through empirical experiments, and then optimize the model using supervised fine-tuning (SFT). For the base model, we select *Qwen2.5-32B-Instruct*, as it has been shown to perform effectively in extensive evaluations.

Although we can distill a large amount of instruction data, we retain only several thousand demonstration instances during SFT. Our ultimate goal is to assess the effectiveness of self-improvement learning within this approach.

3.3 Exploration and Self-Improvement

In this section, we propose enabling LLMs to explore on their own, gradually generating more data for self-improvement.

3.3.1 Iteratively Refined Training Data

We propose an iterative training approach to enhance the slow-thinking capabilities of a model by progressively refining training datasets. The refinement process involves three main strategies: incorporating more accurate trajectories from complex problems, selecting trajectories based on the response length distribution of the same problem type during the imitation phase, and adding high-quality trajectories generated by an improved reasoning model.

Initially, an original dataset, denoted as $\mathcal{D}_0$, containing distilled trajectories from external reasoning systems, is used to train the initial reasoning model. For each problem type T_i in $\mathcal{D}_0$, we calculate the probability distribution of response lengths $p(T_i)$ and assume that the model's self-exploratory answers would also follow this distribution. Once the model is trained, we use it to generate additional trajectories through exploration. Next, we select true trajectories following the corresponding length distribution and add them to $\mathcal{D}_0$ to form a new dataset, $\mathcal{D}_1$. This iterative process of training stronger models and refining training data continues to improve the dataset as the reasoning model evolves. Each refinement step involves strict pre-processing to filter out low-quality trajectories, such as those that are short or noisy. Furthermore, perplexity is identified as a valuable metric for data selection (Ankner et al., 2024), allowing us to identify and retain more challenging trajectories as recognized by the current reasoning model.

3.3.2 Optimization for Self-improvement

After discussing how to iteratively refine training data, we now introduce the optimization methods for self-improvement. We apply two straightforward optimization strategies, integrating the refined training datasets: supervised fine-tuning (SFT) and direct preference optimization (DPO).

When obtaining the refined dataset $\mathcal{D}_t$ in t-th turn, one approach is to iteratively train the base model $\mathcal{M}_{base}$ (*i.e., Qwen2.5-32B-Instruct*) on the refined datasets with SFT to obtain the stronger model $\mathcal{M}_t$, which can be viewed as rejection sampling fine-tuning (Yuan et al., 2023; Zelikman et al., 2022). Another approach to improving the reasoning model is through DPO (Rafailov et al., 2023). This method enhances the model's discrimination capability. During DPO, the model $\mathcal{M}_0$, which is trained on the distilled trajectories (*i.e.,* $\mathcal{D}_0$), is used as the initial model checkpoint. Besides, an SFT loss is incorporated into the objective function to stabilize the training process. This process can be repeated until the dataset is exhausted or a maximum number of iterations is reached.

4 Experimental Analysis and Findings

In this section, we conduct experiments to examine the two key questions: (1) How can LLM learn the slow-thinking mode and (2) How can LLM further improve its reasoning ability without additional demonstration data? In each question, we conduct detailed experiments on two key aspects. For the first question, we want to figure out: (1) What size of backbone model is appropriate during the imitation phase and (2) What is the minimal amount of demonstration data required? For the second question, we want to figure out: (1) How should we iteratively update and refine the exploratory data and (2) How can we balance the final ratios of different data resources effectively? After exploring these questions, we present a final methodology and demonstrate its effectiveness. Our investigation shows that leveraging a small amount of distilled demonstration data can activate the model's long-form thinking capabilities. We also confirm that, even with limited demonstration data, we can achieve comparable results with exploration and self-improvement.

4.1 Experimental Setup

To demonstrate the effectiveness of our framework, we mainly conduct experiments on three challenging benchmarks: MATH-OAI (Lightman et al., 2024), AIME2024 [5], and GPQA (Rein et al., 2023). MATH-OAI contains 500 competition mathematics problems from the MATH (Hendrycks et al., 2021) test set. AIME2024 features 30 problems specifically designed to challenge top high school students with complex problem solving tasks. GPQA consists of 198 multiple-choice problems in biology, physics, and chemistry. In our experiments, we focus on mathematics as the primary domain, with biology, physics, and chemistry serving as auxiliary domains. Among the math benchmarks, MATH-OAI is considered relatively easier, while AIME2024 is regarded as very challenging. Additionally, due to the small number of test samples in AIME2024, its performance tends to fluctuate in our experiments.

We select `Qwen2.5-32B-Instruct` (Team, 2024b) as the backbone model because it demonstrates sufficient foundational capabilities to effectively engage in extended reasoning process. As for baselines, we select several leading o1-like models for comparison (*i.e.,* o1-preview (OpenAI, 2024b), DeepSeek-R1-Lite-Preview (Team, 2024a), and QwQ-32B (Team, 2024c)). In addition, we include GPT-4o (OpenAI, 2024a) and Claude 3.5 Sonnet (Anthropic, 2024), which are advanced general-purpose models. We use greedy search to evaluate the performance of our model with maximum tokens set to 32k.

[5] https://huggingface.co/datasets/AI-MO/aimo-validation-amc

4.2 How to Learn Slow-thinking Mode?

We argue that a well-established model, even with a small amount of long-form thought data, can easily adhere to o1-like output formats. This process is fundamentally about following a prescribed format. The key rationale is that, although the entire thought process may be complex, LLMs can effectively handle individual steps (*e.g.,* planning, verification, and refinement). Therefore, we primarily consider two aspects in activating the model's slow-thinking mode: the scale of model parameters and the volume of long-form complex reasoning thought data.

4.2.1 Sufficient Parameter Size of the Backbone Model is Necessary

Long-thought reasoning is a relatively complex ability, as it often involves a greater number of challenging operations. Therefore, we hypothesize that it imposes certain requirements on the capabilities of the backbone model, which are typically correlated with the model's parameter scale. To identify a suitable model to serve as the backbone, we utilize backbone models of varying sizes (*i.e.,* ranging from 7B to 70B parameters) for comparative analysis. For each model, we train it on the same long-form thought data of 3.9K samples. We show the results in Table 1.

We can find that: firstly, for models with relatively smaller parameters (*e.g.,* 7B), using a limited (*e.g.,* 1.1K) amount of long-thought data can not achieve satisfied performance in the difficult math problems (*i.e.,* Llama-3.1-Instruct-8B only obtain 6.7% performance on AIME, and Qwen2.5-Instruct-7B even can not obtain gains on each task), which is difficult for the later exploration by themselves. Secondly, when the parameter size is increased to 14B, although there is a performance improvement in the field of mathematics (*i.e.,* from 13.3% to 23.3% in AIME), it is still difficult to generalize to other fields (*i.e.,* from 45.5% to 40.9% in GPQA). Thirdly, for models with sufficiently large parameter sizes (*e.g.,* 32B and 70B), a small amount of long-form thought data can trigger the slow-thinking mode, enabling the models to engage in deeper and more meticulous problem-solving

Model Size	Model		MATH-OAI		AIME		GPQA	
	Type	Param.	Acc (%)	Gain (%)	Acc (%)	Gain (%)	Acc (%)	Gain (%)
< 10B	Qwen2.5-Instruct	7B	77.2	-	13.3	-	34.9	-
	*w/*SFT	7B	71.4	-7.5	13.3	0	28.3	-18.9
	Llama-3.1-Instruct	8B	49.2	-	3.3	-	21.7	-
	*w/*SFT	8B	56.2	+14.2	6.7	+100	26.7	+23.0
10B-30B	Qwen2.5-Instruct	14B	79.0	-	13.3	-	45.5	-
	*w/*SFT	14B	80.6	+2.0	23.3	+75.2	40.9	-10.1
30B-70B	Qwen2.5-Instruct	32B	80.0	-	20.0		43.4	-
	*w/*SFT	32B	90.2	+12.8	46.7	**+251.1**	55.1	+22.1
> 70B	Qwen2.5-Instruct	72B	83.1	-	30.0	-	49.0	-
	*w/*SFT	72B	**91.2**	+9.7	**50.0**	+66.7	**59.0**	+19.2
	Llama-3.1-Instruct	70B	65.7	-	6.7	-	44.4	-
	*w/*SFT	70B	81.2	**+23.6**	33.3	**+397.0**	37.9	-14.6

Table 1: Performance comparison of different methods on three representative benchmarks. The **bold** fonts denote the best performance among our training variants, and we report the gain over the backbone model (in percentage).

and achieving performance improvement on both math and other domains (*e.g.,* from 20.0% to 46.7% in AIME and from 43.4% to 55.1% in GPQA).

4.2.2 Limited Amount of Long-form Thought Data is Enough

To explore the amount of long-form thought data required to activate the slow thinking mode in LLMs, we experiment with different data sizes for SFT. We train Qwen-2.5-32B-Instruct using 0.5K, 1K, 2K, 4K, and 6K samples of long-form thought data. We show the results in Table 2. We can see that, even a small amount of data can lead to a significant performance increase (*e.g.,* the average performance has improved from 45.6% to 50.2% with only 0.5K demonstration data). Furthermore, as the volume of demonstration data increases, the model's performance further improves on both math domain and other domains (*e.g.,* from 13.3% to 46.7% in AIME and from 43.4% to 55.0% in GPQA when increasing the number of data from 0K to 4K).

Num.	MATH	AIME	GPQA	Avg.
0K	80.0	13.3	43.4	45.6
0.5K	82.8	23.3	44.4	50.2
1K	86.0	33.3	48.0	55.8
2K	88.2	46.7	52.5	62.5
4K	90.2	46.7	55.0	64.0
6K	89.0	46.7	53.5	63.1

Table 2: Performance comparison with different Long-form Thought data size.

However, we find that beyond a certain point, increasing the data size does not seem to provide additional benefits to the model on the math domain while even hurt the the model in other domains (*e.g.,* same AIME performance and from 55.0% to 53.5% in GPQA when further increasing the number of data from 4K to 6K). Therefore, considering the cost of acquiring long-range reasoning chains, we conclude that 4K data samples are sufficient.

4.3 How to Perform Self-improvement?

In this section, we explore how to leverage the model's self-exploration to achieve results comparable to full imitation learning in scenarios where demonstration data is scarce. Two key aspects are particularly important here: (1) iterative refinement in data selection, and (2) balancing the final model performance by controlling the data miture ratio.

4.3.1 Exploratory Data Should Align with Demonstration Data

During the model's exploration and self-improvement, we iteratively select samples generated by the model itself based on the alignment of response length with the imitation demonstration data. These samples are added to the pool to iteratively expand the volume of exploratory data. To thoroughly demonstrate the importance of length alignment, we conduct experiments using different data length selection strategies. Concretely, we experiment with three selection methods: *align to imitation data* (selecting the trajectories with the same distribution with imitation data that is mentioned in Section 3.3.1, *longest response* (choosing the longest trajectory in the correct trajectories), and *random response* (randomly selecting a correct trajectory).

We present the results of three different selec-

Num.	MATH	AIME	GPQA	Avg.
Align (Ours)	87.4	46.7	53.0	62.4
Longest	86.2	36.7	43.9	55.6
Random	87.2	33.3	48.5	56.3

Table 3: Performance comparison with different selection methods of self-improvement data.

tion methods in the table 3. As the results indicate, aligning the length of exploratory data with the demonstration data is crucial. In this way, we aim to help the model maintain an appropriate level of cognitive effort during exploration. In contrast, random selection and choosing the longest samples lead to significant deviations in the length distribution of the exploratory data, making iterative improvement challenging to achieve.

4.3.2 Challenging Problem is the Key Factor

During SFT training, we prepare a mixture of training data from different domains and varying difficulty levels. In this section, we examine the impact of this data mixture on the model's performance. Specifically, our training dataset consists of three main sources: *hard mathematical problems* (corresponding to difficulty levels such as AIME or the Mathematical Olympiad), *normal mathematical problems* (corresponding to the MATH-OAI difficulty level), and *data from other domains* (corresponding to other disciplines in GPQA). Since the math domain typically contains many challenging reasoning problems, we prioritize it as the main domain.

For the three sources, we experiment with different proportions for data mixture: *w/o hard problems* (removing the hard mathematical problems), *w/o other domains* (removing all non-math data), and *mixed domain data* (including all three parts with a carefully tuned distribution).

We present the performance comparison in Table 5 and derive three major findings. First, excluding the hard problem data leads to a significant drop in performance. This highlights the importance of hard problems in enhancing the reasoning model's capabilities, particularly on the most challenging benchmark, AIME, in our experiments. We observe that hard problems typically require a longer thought process to reach the correct solution (as indicated by the average thought length statistics), which helps better guide and teach LLMs to generate long-form thoughts. Second, using mathematical data alone results in a strong performance across all three benchmarks, not limited to the math domain. This suggests that reasoning with long-form thought is an inherent capability of LLMs, which can be generalized across domains once properly elicited or taught. This finding is particularly significant for the design of generalized reasoning algorithms. Third, introducing a small amount of general data can significantly enhance the model's capabilities in other domains, but it may affect its ability to solve more challenging mathematical tasks. Therefore, how to control the use of general domain data without effecting capabilities in other areas is a promising future direction.

4.4 Final Achieved Results

Here, we show the final performance comparison of various methods on the selected evaluation benchmarks in Table 4. The results include performance metrics for o1-like models, general-purpose models, and several approaches based on the backbone model with additional training methods. We report both the accuracy and the gain relative to the backbone's performance.

From the table (the first part of Table 4), we can observe that industry-level slow-thinking reasoning systems achieve excellent performance across the three benchmarks, showing significant improvement on the most challenging benchmark, *i.e.,* AIME. Secondly, distillation-based variants of our approach (the first group in the second part of Table 4) can yield very competitive results, as shown in the second group of rows, approaching those of industry counterparts (*i.e.,* using 3.9k distilled instances achieves 46.7% and 90.2% accuracy on AIME and MATH-OAI, respectively). Furthermore, increasing the amount of high-quality demonstration data can effectively improve model performance, as evidenced by the comparison between models trained with 1.1k and 3.9k instances. Thirdly, the iteratively trained variants of our approach (the second and third groups in the second part of Table 4) can also achieve promising results across the three benchmarks. Using the variant *w/ SFT 1.1k* as a reference, we observe that incorporating exploration and self-improvement leads to performance improvements for both SFT or DPO, *e.g.,* the performance on AIME goes from 33.3% to 40.0%, 46.7%, and 40.0%, respectively. Additionally, we find that increasing explored instances can also improve the performance to some extent.

Method	Num. Data		MATH-OAI		AIME		GPQA	
	Distill	Explore	Acc (%)	Gain (%)	Acc (%)	Gain (%)	Acc (%)	Gain (%)
GPT-4o	-	-	76.6	-	9.3	-	53.6	-
Claude 3.5 Sonnet	-	-	78.3	-	16.0	-	65.0	-
o1-preview	-	-	85.5	-	44.6	-	72.3	-
DeepSeek-R1-Lite-P	-	-	91.6	-	52.5	-	58.5	-
QwQ-32B-preview	-	-	90.6	-	50.0	-	65.2	-
Backbone	-	-	80.0	-	13.3	-	43.4	-
w/ SFT	3.9k	-	**90.2**	**+12.8**	**46.7**	**+251.1**	55.1	+27.0
w/ SFT	1.1k	-	86.0	+7.5	33.3	+153.8	48.0	+10.6
w/ SFT	1.1k	0.7k	87.1	+8.9	40.0	+200.8	49.0	+12.9
w/ SFT	1.1k	1.6k	87.4	+9.2	**46.7**	**+251.1**	53.0	+22.1
w/ SFT	1.1k	1.8k	89.8	+12.3	40.0	+200.8	**56.1**	**+29.3**
w/ SFT & DPO	1.1k	0.3k	87.2	+9.0	30.0	+125.6	49.5	+14.1
w/ SFT & DPO	1.1k	1.0k	85.4	+6.8	**46.7**	**+251.1**	51.0	+17.5

Table 4: Performance comparison of different methods on three representative benchmarks. "Backbone" refers to CoT reasoning method based on the `Qwen2.5-32B-Instruct` model, while "*w/* SFT" and "*w/* SFT & DPO" denote training with our proposed method. The columns of "Distill" and "Explore" indicate that the source of training instances, either distillation from R1 and QwQ or exploration by the model itself. The **bold** fonts denote the best performance among our training variants, and we report the gain over the backbone model (in percentage).

Settings	Avg. L	MATH	AIME	GPQA	Avg.
w/o HP	2866	86.0	33.3	51.0	56.8
w/o OD	3389	87.4	46.7	53.0	62.4
mixed	3162	89.8	40.0	56.1	62.0

Table 5: Performance comparison with different mixtures for multi-domain data. We also report the average length for each data mixture.

Empirically, we find that the improvement of iterative training is often limited to the initial iterations and might lead to performance fluctuations on some benchmarks. We speculate that, due to the constrained number of rollouts (at most 20 in our experiments), a portion of challenging problems cannot be correctly solved by our reasoning model, which can be solved by increasing the rollout number.

Overall, our distillation-based variant (with 3.9k instances) achieves the best performance among all our attempts, approaching the performance of industry-level reasoning systems. Meanwhile, the variants incorporating exploration and self-improvement also show substantial improvements over the backbone model.

5 Conclusion

In this paper, we present a detailed introduction to a reproduced o1-like reasoning system. We outline a two-phase development approach for implementing such a capable system, where the model is initially trained using distilled long-form thought data and then undergoes self-improvement by exploring difficult problems. Our system has demonstrated strong performance on three challenging evaluation benchmarks. We find that the slow-thinking mode can be easily transferred across domains and is particularly effective at solving hard, complex problems. Our main findings can be summarized as follows:

(1) The ability to perform long-form thinking can be effectively elicited by training with a small amount of high-quality demonstration data. Once established, this ability appears to naturally generalize across domains.

(2) Demonstration data from the math domain is particularly well-suited for developing the long-form thinking ability of LLMs, and data with longer thought processes appears especially effective in enhancing the model's capacity to tackle challenging problems.

(3) Unlike the formal responses generated by LLMs in a fast-thinking mode, the thought process is typically expressed in a flexible, informal manner, serving to guide LLMs toward the correct path to the solution.

(4) The slow-thinking capability can be effectively enhanced through exploration and self-improvement, whereas the improvements from offline learning methods seem to occur primarily in the initial stage, especially for challenging tasks.

Limitations

Despite the promising results, our exploration remains preliminary, and there is still a substantial capacity gap compared to industry-level systems. In addition to the two methods described above, another promising training approach is reinforcement learning (Schulman et al., 2017; Ye et al., 2024), where the policy model is directly trained during the exploration process. However, due to computational resource constraints, we leave this approach for future work. As future work, we plan to investigate how to scale our training approach and extend its capacity to more complex tasks.

References

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L. Leavitt, and Mansheej Paul. 2024. Perplexed by perplexity: Perplexity-based data pruning with small reference models. *CoRR*, abs/2405.20541.

Anthropic. 2024. Claude 3.5 sonnet.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Xin Zhao, and Ji-Rong Wen. 2023. Chatcot: Tool-augmented chain-of-thought reasoning on chat-based large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 14777–14790. Association for Computational Linguistics.

Zhipeng Chen, Kun Zhou, Xin Zhao, Junchen Wan, Fuzheng Zhang, Di Zhang, and Ji-Rong Wen. 2024. Improving large language models via fine-grained reinforcement learning with minimum editing constraint. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 5694–5711. Association for Computational Linguistics.

Kahneman Daniel. 2017. *Thinking, fast and slow*.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *CoRR*, abs/2311.04205.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. PAL: program-aided language models. *CoRR*, abs/2211.10435.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. CRITIC: large language models can self-correct with tool-interactive critiquing. *CoRR*, abs/2305.11738.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1051–1068. Association for Computational Linguistics.

Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024. O1 replication journey–part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *arXiv preprint arXiv:2411.16489*.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, Zheng Liu, Dong Yan, Jian Xie, Zhongyuan Wang, and Ji-Rong Wen. 2024a. Technical report: Enhancing llm reasoning with reward-guided tree search.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, et al. 2024b. Technical report: Enhancing llm reasoning with reward-guided tree search. *CoRR*, abs/2411.11694.

D. Kahneman. 2011. Thinking, slow and fast.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *ICLR*. OpenReview.net.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, Xin Jiang, and Qun Liu. 2024. Self: Self-evolution with language feedback.

OpenAI. 2024a. Hello gpt-4o.

OpenAI. 2024b. Learning to reason with llms.

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. *CoRR*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

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

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

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *CoRR*, abs/2404.14387.

DeepSeek Team. 2024a. Deepseek-r1-lite-preview is now live: unleashing supercharged reasoning power!

Qwen Team. 2024b. Qwen2.5: A party of foundation models.

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

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M. Ni, Linyi Yang, Ying Wen, and Weinan Zhang. 2024a. Openr: An open source framework for advanced reasoning with large language models. *CoRR*, abs/2410.09671.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. 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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

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

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

Jason Weston and Sainbayar Sukhbaatar. 2023. System 2 attention (is something you might need too). *CoRR*, abs/2311.11829.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Chenlu Ye, Wei Xiong, Yuheng Zhang, Nan Jiang, and Tong Zhang. 2024. Online iterative reinforcement learning from human feedback with general preference model. *CoRR*, abs/2402.07314.

Dongran Yu, Bo Yang, Dayou Liu, Hui Wang, and Shirui Pan. 2023. A survey on neural-symbolic learning systems. *Neural Networks*, 166:105–126.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *CoRR*, abs/2407.06023.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: LLM self-training via process reward guided tree search. *CoRR*, abs/2406.03816.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, Wanli Ouyang, and Dongzhan Zhou. 2024b. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *CoRR*, abs/2410.02884.

Kaiyan Zhang, Biqing Qi, and Bowen Zhou. 2024c. Towards building specialized generalist AI with system 1 and system 2 fusion. *CoRR*, abs/2407.08642.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *CoRR*, abs/2411.14405.

Appendix

A Long-form Thought Dataset Construction

Data Collection. In practice, there are three typical approaches to constructing long-form thought data. First, human annotators can be employed to generate this data. Second, LLMs can be employed generate long-form thought data with the assistance of auxiliary search algorithms (*e.g.,* Monte Carlo Tree Search). Third, this data can be distilled from o1-like reasoning systems. Given considerations of simplicity and budget constraints, we adopt the third approach for collecting long-form thought data, recognizing that our approach can be extended to include other data collection methods. We access two o1-like reasoning systems: `DeepSeek-R1-Lite-Preview` (Team, 2024a) (abbreviated as *R1*) and `QwQ-32B-preview` (Team, 2024c) (abbreviated as *QwQ*). R1 provides an open API with limited access, while QwQ offers open model checkpoints. We opt not to distill data from o1, as it summarizes internal thought data before presenting it to the user, rendering it not directly suitable for imitation. Upon examining the thought processes of R1 and QwQ, we find that the thoughts can be presented in a relatively flexible and even colloquial manner. To obtain the distilled data, we input the specified problems into both systems and perform multiple rollouts to generate a diverse set of responses. We then automatically label each solution based on whether it leads to the correct answer.

Format Unification. After examining the thought processes of R1 and QwQ, we observe that R1 explicitly separates the thinking and solution components, whereas QwQ directly outputs the correct answer following the thinking process. A comparative example of the two systems is shown in Figure 2. Following R1's structure, we divide the response into two parts, separated by the tokens "`begin/end_of_thought`" and "`begin/end_of_solution`".

Since QwQ does not explicitly contain a solution component, we consider performing a rollout to complete this part. Specifically, we first fine-tune a reasoning model (either QwQ or our model) with the distilled, formatted data from R1 and then prompt it to complete the solution section. We find

Problem: Given a triangle ABC with side lengths a, b, and c, an inradius of 3, and a circumradius of 7, find the value of $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac}$

QwQ

- So I've got this problem about a triangle with sides a, b and c. It says the inradius is 3 and circumradius is 7, and I need to find the value of $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac}$. Hmm, I'm not super familiar with inradius and circumradius, but I know that the inradius is the radius of the inscribed circle, and the circumradius is the radius of the circumscribed circle. Maybe I can find some formulas that relate these to the sides of the triangle.
- First, I recall that the area of a triangle can be expressed in terms of its inradius and its semiperimeter. The semiperimeter, usually denoted by s, is half the perimeter, $s = \frac{a+b+c}{2}$. The area, A, is then $A = r \cdot s$, where r is the inradius. So in this case, $A = 3s$.
- I also know that the area of a triangle can be expressed using its circumradius, R. There's a formula that says $A = \frac{abc}{4R}$. Given that R is 7, that would make $A = \frac{abc}{28}$.
- So now I have two expressions for the area: $A = 3s$ and $A = \frac{abc}{28}$. Maybe I can set them equal to each other: $3s = \frac{abc}{28}$. That gives me $abc = 84s$. Hmm, that might be useful later.
- I need to find $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac}$. Let's see, that's equal to $\frac{c+b+a}{abc}$, which is $\frac{a+b+c}{abc}$. But since $s = \frac{a+b+c}{2}$, then $a+b+c = 2s$. So, $\frac{a+b+c}{abc} = \frac{2s}{abc}$.
- Let me double-check my steps. First, area $A = 3s$ and $A = \frac{abc}{28}$, so setting them equal: $3s = \frac{abc}{28}$, which gives $abc = 84s$. Then, $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac} = \frac{a+b+c}{abc} = \frac{2s}{84s} = \frac{1}{42}$. Yeah, that seems correct.
- But just to be thorough, let's see if there's another way to approach this. maybe I can use Heron's formula for the area in terms of the sides, but that seems more complicated than what I've already done.
- Alternatively, perhaps I can use the fact that in any triangle, $r(R + r) = \sqrt[3]{abc}$, but I'm not sure about that.
- Wait, perhaps I should consider the expression ($\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac} = \frac{c+a+b}{abc} = \frac{a+b+c}{abc}$) I'm trying to find. I already have $a+b+c = 2s$ and $abc = 84s$, so $\frac{a+b+c}{abc} = \frac{2s}{84s} = \frac{1}{42}$. It seems consistent with what I got earlier.
- Alternatively, maybe I can consider the expression $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac} = \frac{1}{abc} \cdot (a+b+c)$, which is again $\frac{2s}{abc}$, and with $aabc = 84s$, it simplifies to $\frac{2s}{84s} = \frac{1}{42}$.
- I think I've confirmed it in multiple ways now. So, the answer should be $\frac{1}{42}$.
- **Final Answer**

$$\frac{1}{42}$$

Deepseek

Thought:

- So I've got this problem here. We have a triangle ABC with sides a, b, c, an inradius of 3, and a circumradius of 7. I need to find the value of $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac}$. Hmm, okay, let's break this down step by step.
- First off, I know that in a triangle, the inradius (r) and the circumradius (r) are related to other elements of the triangle, like the area and the sides. I recall that the area of the triangle can be expressed in terms of the inradius and the semiperimeter: area $= r \cdot s$, where s is the semiperimeter, $s = \frac{a+b+c}{2}$.
- Also, the area can be expressed in terms of the circumradius: area $= \frac{abc}{4r}$. So, since both expressions equal the area, I can set them equal to each other: $r \cdot s = \frac{abc}{4r}$. Plugging in the known values, r = 3 and $r = 7$, so: $3s = \frac{abc}{4\times7} \Longrightarrow 3s = \frac{abc}{28} \Longrightarrow abc = 84s$.
- Okay, that's one equation involving a, b, c, and now, I need to find $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac}$. Let me see if I can express this in terms of s or other known quantities. First, $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac}$ can be written as $\frac{c+b+a}{abc}$. Since $a+b+c = 2s$, this becomes $\frac{c+b+a}{abc}$. From earlier, I have $abc = 84s$, so plugging that in: $\frac{2s}{84s} = \frac{2}{84} = \frac{1}{42}$.
- Alternatively, I could use the formula for the area in terms of r and r and see if that leads me somewhere, but I think the approach I took is valid. Let me see if I can confirm this with another method. Perhaps using the formula for the area in terms of r and s. I know that area $= r \cdot s = 3s$, and area $= \frac{abc}{4r} = \frac{abc}{28}$. Setting them equal: $3s = \frac{abc}{28} \Longrightarrow abc = 84s$. Now, back to $\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac} = \frac{c+b+a}{abc} = \frac{2s}{abc}$. Substituting $abc = 84s$, we get $\frac{2s}{84s} = \frac{2}{84} = \frac{1}{42}$.

Solution:

- First, we use the fact that the area of the triangle can be expressed in terms of the inradius and semiperimeter:
$$\delta = r \cdot s = 3s$$
- Where $s = \frac{a+b+c}{2}$ is the semiperimeter.
- Additionally, the area can be expressed using the circumradius:
$$\delta = \frac{abc}{4r} = \frac{abc}{28}$$
- Equating the two expressions for the area:
$$3s = \frac{abc}{28} \Longrightarrow abc = 84s$$
- Next, we need to find:
$$\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac} = \frac{c+a+b}{2}$$
- Since $a+b+c = 2s$, we have:
$$\frac{1}{ab}+\frac{1}{bc}+\frac{1}{ac} = \frac{2s}{abc}$$
- Substituting $abc = 84s$:
$$\frac{2s}{84s} = \frac{2}{84} = \frac{1}{42}$$
$$3s = \frac{abc}{28}$$
- Thus, the value is :
$$\frac{1}{42}$$

Figure 2: A case study comparing QwQ with DeepSeek in solving math problems.

that, given the preceding thought process, the reasoning model can readily generate the solution if trained using imitation learning. The final format of our demonstration data is shown below:

Long-form Thought Format for Our Reasoning Model

```
<|begin_of_thought|>
{different step of thought separated by \n\n}
<|end_of_thought|>

<|begin_of_solution|>
{formated step-by-step final solution}
<|end_of_solution|>
```

Prompt Template for Our Reasoning Model

```
Your role as an assistant involves thoroughly
exploring questions through a systematic
long thinking process before providing
the final precise and accurate solutions.
This requires engaging in a comprehensive
cycle of analysis, summarizing, exploration,
reassessment, reflection, backtracing, and
iteration to develop well-considered thinking
process.

Please structure your response into two
main sections:  Thought and Solution.

In the Thought section, detail your reasoning
process using the specified format:
“‘
<|begin_of_thought|>
{thought with steps separated with "\n\n"}
<|end_of_thought|>
”’
Each step should include detailed
considerations such as analisying questions,
summarizing relevant findings, brainstorming
new ideas, verifying the accuracy of the
current steps, refining any errors, and
revisiting previous steps.

In the Solution section, based on various
attempts, explorations, and reflections
from the Thought section, systematically
present the final solution that you deem
correct.  The solution should remain a
logical, accurate, concise expression style
and detail necessary step needed to reach the
conclusion, formatted as follows:
“‘
<|begin_of_solution|>
{final formatted, precise, and clear
solution}
<|end_of_solution|>
”’
Now, try to solve the following question
through the above guidelines:
```

Data Mixing. Our goal is to develop more generalized LLMs capable of reasoning across different domains. To achieve this, we begin by using demonstration instances (problems paired with their distilled responses) from mathematic domain, as we hypothesize that the ability to perform long-form reasoning can transfer easily across them. The second consideration is the difficulty of the demonstration instances. Intuitively, applying long-form reasoning to solve relatively simple problems may be less beneficial. Therefore, we focus on collecting more challenging problems from the selected domains. Specifically, we select problems from the MATH and Olympiads subsets of the Numina-MATH (Li et al., 2024) dataset, as well as AIME problems collected from the AOPS website [6] spanning 1983 to 2023.

Pre-processing Demonstration Data. After collecting the labeled data, we perform further pre-processing to ensure data quality, including deduplication and filtering. Specifically, when generating long-form thought, existing models often produce issues such as repetitions, gibberish, or mixtures of English and Chinese. To address this, we use rule-based methods (*e.g.,* regex matching and n-gram matching) to remove such instances. Another key observation is that longer instances tend to lead to better performance, so we also remove relatively short examples. As a result, we obtain a cleaned demonstration dataset suitable for fine-tuning our reasoning model. Additionally, we employ the following prompt to guide the model in performing slow thinking more effectively.

[6] https://artofproblemsolving.com