One Missing Piece for Open-Source Reasoning Models: A Dataset to Mitigate Cold-Starting Short CoT LLMs in RL

Hyungjoo Chae[1,2,*]**, Dongjin Kang**[1,2,*]**, Jihyuk Kim**[2]**, Beong-woo Kwak**[1]**,**
Sunghyun Park[2]**, Haeju Park**[2]**, Jinyoung Yeo**[1]**, Moontae Lee**[2,3]**, Kyungjae Lee**[2]

[1]Yonsei University [2]LG AI Research [3]University of Illinois Chicago
{mapoout, hard1010, jinyeo}@yonsei.ac.kr
{moontae.lee, kyungjae.lee}@lgresearch.ai

Abstract

With the release of R1, a publicly available large reasoning model (LRM), researchers commonly train new LRMs by training language models on R1's long chain-of-thought (CoT) inferences. While prior works show that LRMs' capabilities can be reproduced through direct distillation, the continued reliance on the existing models (e.g., R1) remains a critical limitation in advancing the field. As a first step toward independent LRM development, this paper explores the possibility of constructing a long CoT dataset with LLMs that are not trained for inference-time scaling. To this end, we present the Long CoT Collection, a dataset of 100K CoT rationales annotated using existing short CoT LLMs. We develop a pipeline that induces o1's novel reasoning strategies into short CoT LLMs, enabling them to think longer and introducing controllability over the thought budget to better manage the overthinking problem. Our extensive analyses validate that our dataset achieves quality comparable to—or slightly below—R1. Furthermore, our experiments demonstrate that training on our dataset not only strengthens general reasoning skills, but also provides a strong foundation for reinforcement learning—models initialized on our data achieve 2-3x larger gains with RLVR. We make the codes, datasets, and models publicly available at LINK.

1 Introduction

Large Reasoning Models (LRMs), exemplified by the o-series (OpenAI, 2024), have shown groundbreaking performance in various reasoning tasks with test-time scaling (i.e., generating extremely long chain-of-thought (CoT) rationales). (Guan et al., 2025; Zhang et al., 2024b; Yu et al., 2025). However, their closed nature presents significant challenges—its high API costs and safety issues
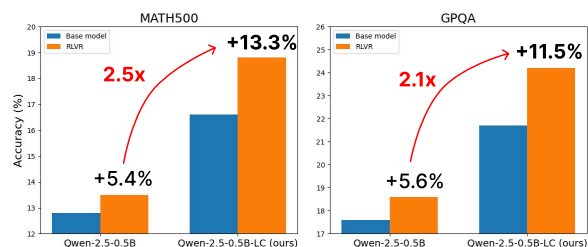

Figure 1: Comparison of RLVR performance between the base model (Qwen-2.5-0.5B) and the model trained on the Long CoT Collection (Qwen-2.5-0.5B-LC) on MATH500 and GPQA.

limit real-world applications (Hendrycks et al., 2022), while the closed-source approach potentially prohibits academic progress in the field.

To address these issues, DeepSeek-AI et al. (2025) release an open-source version of o1 and detail their methodology for building R1. While the benefits of reinforcement learning with verifiable reward (RLVR) have been previously demonstrated (Lambert et al., 2024), they introduce a key innovation by tackling the cold-start instability in RL training for Short CoT LLMs. Finetuning on a carefully curated Long CoT dataset to explicitly teach reasoning structures serves as a critical step to enable the model to acquire the foundational reasoning skills before RL. Building on this insight, subsequent works have shown that simply collecting R1's outputs to construct a Long CoT dataset and fine-tuning LLMs on it can lead to dramatic improvements (Labs, 2025; Team, 2025b). Furthermore, Yeo et al. (2025) provide a detailed analysis of the role of RLVR following this finetuning stage.

Yet, despite these advancements, an important gap remains: the cold-start problem itself has not been fully demystified. While R1's Long CoT dataset serves as a critical ingredient, the exact mechanisms for creating such data have remained unclear. In this work, we investigate whether it is possible to construct Long CoT data from the short

---

CoT responses of LLMs that have been trained to produce only concise rationales. Instead of directly collecting LRMs' completions, we built a simple pipeline that enables LLMs to generate long CoT in a step-by-step manner with only a small guidance from LRMs. To allow LLMs to annotate long CoT, we begin by creating a seed dataset of 1K instances, capturing o1's reasoning flow that reflects its novel reasoning strategies. Then, we generate the reasoning flow on the new question and expand it to long CoT with short CoT LLMs (e.g., GPT-4o) in a step-by-step manner. The resulting collection of 100K instances serves as a comprehensive training resource, allowing base LLMs to learn to think longer while incorporating diverse reasoning strategies characteristic of o1. Since this collection process offers controllability over the thought budget, it has a strong advantage in addressing one of the major issues with LRMs: overthinking—generating an unnecessarily large number of tokens for simple problems.

To further validate our approach, we conduct in-depth analyses of the quality of our dataset. Despite being generated by short CoT LLMs, the rationales in our dataset demonstrate reasoning flows and strategies that nearly match the quality of R1 in terms of reasoning flow, showing only slightly lower performance in other criteria. In addition, the generated rationals contain rich reasoning triggers (e.g., "Wait" and "To verify") that help explore diverse reasoning paths and enhance accuracy. Our thought budget analysis shows that short CoT LLMs, guided by the example reasoning flow, effectively allocate their computational resources in alignment with state-of-the-art reasoning models.

Through extensive experiments, we demonstrate that the Long CoT Collection provides an effective foundation for initializing SFT models for reinforcement learning (RL). Best-of-n sampling comparisons show that models trained on our dataset consistently outperform the base models, demonstrating strong potential when optimized for outcome-based rewards. Evaluations on GPQA (Rein et al., 2023) and MMLU-Pro (Wang et al., 2024b) further highlight that training on our dataset enhances reasoning capabilities across general domain tasks. Notably, initializing policies with our dataset before RL leads to 2-3x greater performance improvements, demonstrating out collection's strong potential to accelerate and stabilize downstream learning (Figure 1).

2 Related Work

Inference-time Scaling. Recent research has demonstrated that scaling inference-time improves efficiency and overall reasoning quality by increasing the number of tokens, compared to traditional scaling laws such as increasing model parameters or dataset volumns (Brown et al., 2024; Snell et al., 2024). This can be achieved by sampling many reasoning paths (e.g., Best-of-N (Snell et al., 2024) and MCTS (Zhang et al., 2024a)) and using a verifier or voting mechanism to pick the correct solution (e.g., self-consistency) (Liang et al., 2024). Furthermore, OpenAI (2024); DeepSeek-AI et al. (2025) explore training LLMs to generate a long CoT, similar to how humans handle complex tasks, which often involve self-correction or verification before arriving at a final answer. This shift towards deliberative reasoning makes LLMs more transparent, interpretable, and adaptable in complex decision-making scenarios (Yeo et al., 2025).

Large Reasoning Models and Datasets. Since the success of OpenAI's o1 model (OpenAI, 2024), many studies have attempted to replicate o1-like reasoning as open-source models (Team, 2024, 2025a; Muennighoff et al., 2025b). Recent studies emphasize the importance of the dataset used for initializing these LRMs (Xu et al., 2025; Muennighoff et al., 2025b; Ye et al., 2025). Notably, DeepSeek-AI et al. (2025) demonstrated that introducing a brief supervised fine-tuning (SFT) stage—where the model is "cold-started" with a few thousand high-quality CoT examples—leads to a more stable and efficient RL stage. High-quality SFT datasets for reasoning are thus a key ingredient for these models, yet current public datasets remain limited. To compensate, researchers have begun curating their own reasoning corpora (Guan et al., 2025; Xu et al., 2025; Pang et al., 2025; Ye et al., 2025). To address this critical gap, we introduce the Long CoT Collection, a large-scale dataset specifically designed to initialize models for complex reasoning tasks through supervised fine-tuning.

Reinforcement Learning for Reasoning. Reinforcement learning with human feedback (RLHF) has become a dominant paradigm for aligning LLMs to human preferences (Ouyang et al., 2022; Bai et al., 2022; Touvron et al., 2023). In RLHF, a reward model that learns human preference guides the policy to produce responses that humans would rate highly (e.g., helpful and harmless re-
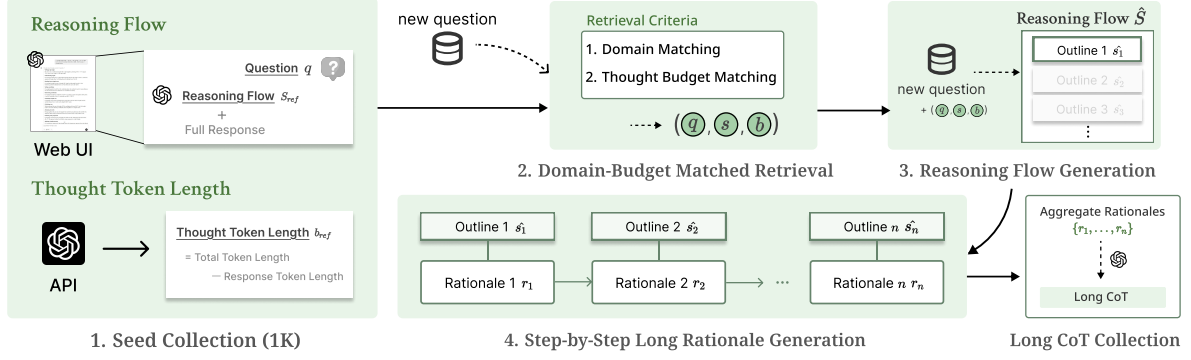
Figure 2: Overview of our data construction pipeline. First, we collect an 1K seed dataset of reasoning flow and thought token length (1). Using it as a demonstration, we annotate long CoT rationales on new questions and scale it up to 100K data points (2-4).

sponses) (Zhu et al., 2023). However, Lambert et al. (2024) have pointed out that relying on a learned reward model can introduce instability in the RL process. To tackle this, researchers are turning to RLVR as a more grounded alternative for reasoning domains (Lambert et al., 2024; DeepSeek-AI et al., 2025). The idea is to focus on objective, checkable outcomes rather than learning a proxy for human preferences, providing rewards only when its output is correct.

3 The Long CoT Collection

In this section, we present the Long CoT Collection, a dataset for learning LRMs' emergent reasoning behavior. To allow more openness and controllability of the data collection process, we investigate whether long CoT data can be annotated by short CoT LLMs. Our data collection process begins by collecting 1K demonstrations that capture LRMs' reasoning flow (Section 3.1), then generating 100K long CoT data using short CoT LLMs guided by the seed demonstrations (Section 3.2). The overall construction process is illustrated in Figure 2.

3.1 Collecting Teacher Demonstrations

A key challenge in building long CoT datasets with short CoT LLMs is allowing them to generate long rationales with coherence. To address this, we first collect a seed dataset with o1 that reflects the novel reasoning process of LRMs.

3.1.1 Reasoning Flow Annotation

Reasoning flow S is an overview of the reasoning process that consists of a sequence of outlines $\{s_1, s_2, ..., s_n\}$ for each reasoning step. It contains crucial information about the reasoning process and how the logical steps flow from the initial problem

understanding to the final conclusion. We manually collect reference reasoning flow S_{ref} from ChatGPT website, using the question q from 1K reasoning-focused instructions from the magpie-reasoning-V1 dataset (Xu et al., 2024). In addition, our dataset includes thought budget b_{ref} (i.e., the number of thought tokens used) of o1 by calculating the difference between the total completion token count and the number of tokens in the returned response, using the OpenAI API. As a result, we collect 1K seed dataset $\mathcal{D}_{ref} \in \{q, S_{ref}, b_{ref}\}$ that will be used in Section 3.2. We show the distribution of the title of the reasoning outline in Figure 3.

3.2 Annotating Long CoT with Indirect Guidance from Teacher

Using the 1K seed dataset as our foundation, we expand it to 100K data. Since short CoT LLMs struggle to maintain coherence during extended test-time computing, we breakdown the reasoning into three steps to enable step-by-step generation of long CoT rationales.

3.2.1 Reasoning Flow Retrieval

Each question has its own reasoning procedure to reach the answer. Thus, for the new question q, we dynamically retrieve demonstrations (q, S_{ref}, b_{ref}) from our seed dataset $\mathcal{D}_{ref}$ to teach LLMs to generate reasoning flow S with in-context learning. The following aspects are considered for the retrieval: **(1) Domain matching**: Problems in the same or similar domain are highly likely to share a common reasoning process. For example, in arithmetic reasoning, o1 tends to verify its calculation to ensure the correct answer. We use the primary domain and sub-domain in the magpie-V1-reasoning dataset to calculate the domain matching score (Xu et al., 2024). **(2) Thought budget control**: To align with
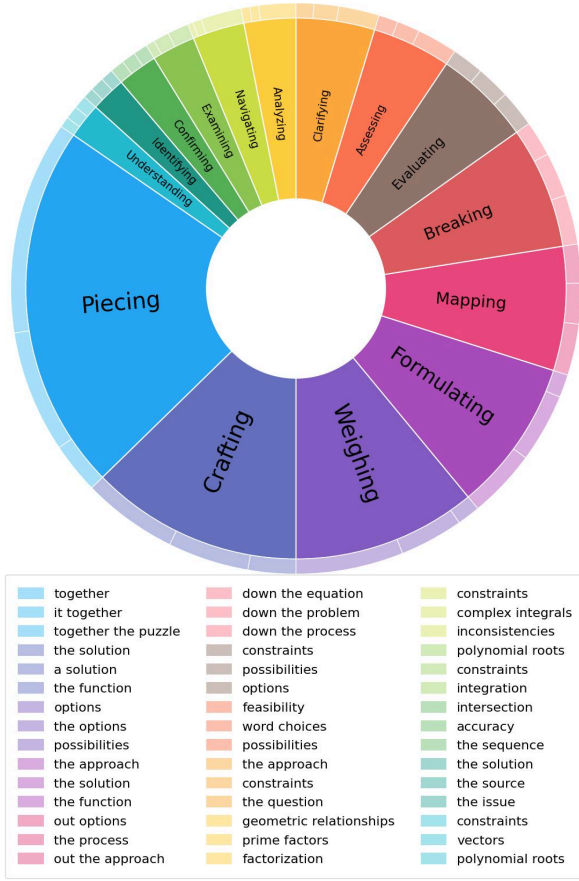
Figure 3: The top 15 most common root verbs and their top 3 direct noun objects in the collected reasoning flow.

Figure 4: Head-to-head comparison of the generated CoT quality with the R1 output (Ye et al., 2025).

3.2.3 Step-by-step Long CoT Generation with Reasoning Flow

Using the generated reasoning flow $\hat{S}$ as guidance, LLMs generate long CoT rationale step-by-step. Specifically, for each step $\hat{s}_i$ in $\hat{S}$ LLMs generate rationales r_i based on the given previous reasoning $\{r_k\}_0^{i-1}$, the current flow step $\hat{s}_i$, and the next flow step $\hat{s}_{i+1}$. When the summary steps are all consumed, the LLMs generate the final solution based on the reasoning. At last, the reasoning steps and the final answer are aggregated as a sequence.

3.2.4 Correctness Filtering

Lastly, we filter out the rationales that results in wrong answers, as training on incorrect rationales might harm their original reasoning capability. Specifically, we simply ask GPT-4o to validate the answer given the reference answer and the generated answer span. This filtering results in 76% instances with correct answer prediction.

4 Dataset Analyses

4.1 High Quality

We focus on three important aspects; (1) **Reasoning Flow**: The logical progression and coherence of steps in the solution process, measuring how naturally one step leads to the next. (2) **Reasoning Strategy**: The specific techniques and approaches employed to break down and solve problems, such as the selection of relevant mathematical tools or problem-solving methods. (3) **Correctness**: The accuracy of each reasoning steps.

We compare our method with a widely used method for long CoT data generation which collects the outputs from the existing LRMs. For a fair comparison, we sample 100 questions from the Long CoT Collection for which R1-generated solutions have the correct answer. Following the finding that stronger policy models can be used for trajectory scoring (Wang et al., 2024a), we use

reference LRMs, the thought budget is controlled by retrieving reasoning flows of similar length for demonstration. We measure this similarity using $1 - \left| \frac{\min(x,y)}{\max(x,y)} - 1 \right|$, where x and y represent the reference and candidate budgets, respectively. The heatmap of this similarity function is in Figure 13.

3.2.2 Reasoning Flow Generation

The retrieved demonstrations teach LLMs, which is GPT-4o in our experiment, to imagine LRMs' reasoning behavior at a higher level. Without the demonstration, we find that LLMs only stick to a linear thinking process, where the reasoning proceeds in one direction and does not include LRMs' novel reasoning strategies, such as verification and exploration of diverse solutions. LLMs generate reasoning flow $\hat{S}$ on the new question, given the retrieved demonstration. Specifically, they first predict the expected number of outlines $|S|$ and generate a sequence of reasoning outlines that emulates the higher-level reasoning patterns observed in the retrieved demonstrations.
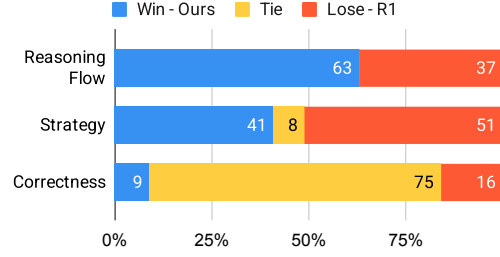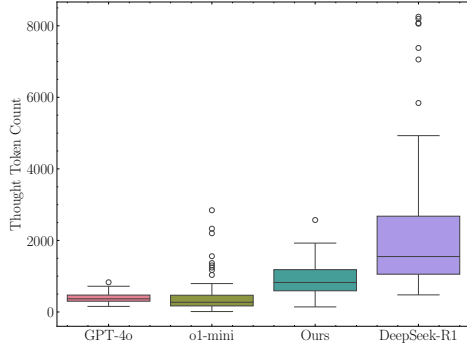
Figure 5: Comparison of the number of reasoning tokens used by each model.

the state-of-the-art LRM, o3-mini, as our evaluator. Figure 4 shows that the rationales from the Long CoT Collection demonstrate better reasoning flow, and while showing slightly weaker strategy and correctness, they remain competitive.

4.2 Efficient Thought Budget Allocation

Allocating the proper budget for thinking is an important issue (Wang et al., 2025). LRMs tend to use too many thought tokens for easy problems (i.e., overthinking), which leads to a huge amount of computational cost. To evaluate the efficiency in thought token allocation, we analyze the rationale lengths and compare them against other LRMs and GPT-4o, the LLM used in constructing our dataset. Specifically, we randomly sample 100 instances from the Long CoT Collection and annotate the rationales with each model. As Figure 5 indicates, simple CoT prompting on GPT-4o rarely generates rationales longer than 1,000 tokens, which suggests that naive prompting on GPT-4o is hard to use for constructing long CoT datasets. In addition, R1 uses significantly more thought tokens than o1-mini, which results in overthinking when models are trained on its outputs.

5 Effect of the Long CoT Collection

As demonstrated in prior works (DeepSeek-AI et al., 2025; Yeo et al., 2025), the training of LRMs typically follows a two-phase approach: first, imitation learning to master long-form CoT reasoning, followed by RL to enhance reasoning accuracy. In this section, we investigate the impact of training LLMs on our dataset from two perspectives: its effectiveness as a starting point for RL and its actual impact on the RL training phase.

5.1 A Reliable Starting Point for RL

Setup. RL for inference-time scaling includes sampling trajectories from the policy model and updating the policy based on calculated rewards. In such sparse reward settings, the quality of the initial policy model is critical—if the model rarely generates high-reward trajectories at the start, the learning signal may be too weak for effective training. To assess the potential of our initial policy model, we evaluate its performance using best-of-n (BoN) sampling, which reveals the model's capacity to generate correct solutions when allowed multiple attempts. We assess our model on mathematical reasoning benchmarks, as they widely used for RL to elicit inference-time scaling. We choose two challenging benchmarks, MATH-500 (Lightman et al., 2023) and AIME24 (of America, 2024). We use Llama-3.1-8B-Instruct (Dubey et al., 2024) and Qwen2.5-7B-Instruct (Qwen et al., 2024) as our base model. We train them on our dataset, and the full hyperparameters are in Appendix B.2.

Results. Figure 6 shows BoN results with two base models. We measure Pass@N (N=1,2,4,8,16 and 32), where a set of N samples is considered correct if at least one sample includes the ground-truth answer. On Llama-3.1-8B-Instruct, we observe notable improvement on both benchmarks, consistently across different N. Meanwhile, our Qwen2.5-7B-LC improves performance given large N (e.g., 16 or 32), while the performance of Qwen2.5-7B-Instruct quickly saturates. This shows that our SFT training recipe enables the model to explore more diverse responses and thus leads to higher answer reward when applied to RL.

5.2 Impact on General Reasoning Domains

Setup. Along with the mathematical benchmarks, we test our model on the general reasoning benchmarks, GPQA Diamond (Rein et al., 2023) and MMLU-Pro (Wang et al., 2024b) (see Appendix B.4 for details). We consider the baselines in the following three categories; (1) Closed-source LRMs: OpenAI's o1 and o1-mini (OpenAI, 2024) demonstrate state-of-the-art performance but are accessible only through APIs. (2) Open-source LRMs with undisclosed SFT datasets: R1 (DeepSeek-AI et al., 2025) and QwQ (Team, 2024) successfully replicate o1's capabilities, but the datasets for SFT remain undisclosed. (3) Open-source LRMs via distillation: Models like Sky-T1 (Team, 2025a) and Bespoke-7B (Labs, 2025) utilize open-source
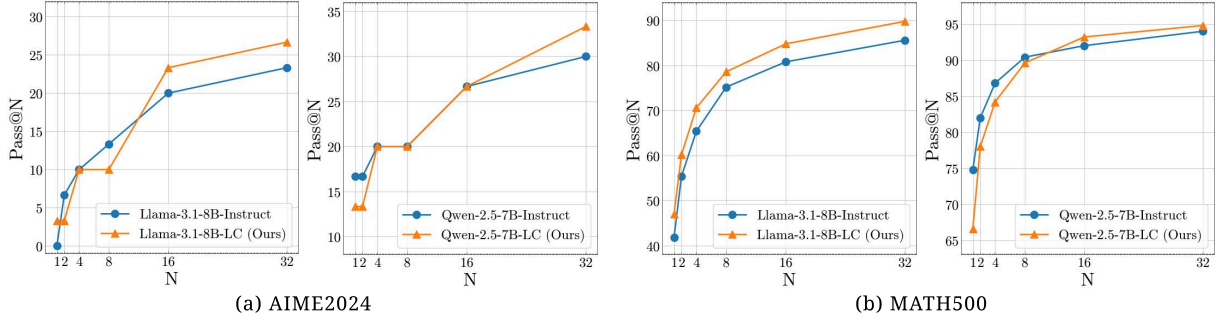
Figure 6: Results of best-of-n experiments with Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct.

Model	Size	GPQA Diamond	MMLU Pro
API only			
o1-mini	N/A	60.0	80.3
o1	N/A	77.3	-
Open Weights			
Qwen-2.5-32B-Instruct	32B	49.0	69.2
QwQ-32B	32B	65.2	71.0
R1	671B	71.5	84.0
Qwen-7B-R1-distill	7B	49.1	-
Open Weights and Open Data			
Sky-T1	32B	56.8	69.2
Bespoke-7B	7B	38.9	-
OpenThinker-7B	7B	42.4	-
Llama-3.1-8B-Instruct	8B	22.7	43.7
Llama-3.1-8B-LC (Ours)	8B	36.4	44.5
Qwen-2.5-7B-Instruct	7B	37.6	49.9
Qwen-2.5-7B-LC (Ours)	7B	39.9	51.4

Table 1: Performance of various reasoning models. Some results are from the respective reports

datasets collected from existing LRMs' outputs.

Results. We present our results in Table 1. Models trained on the Long CoT Collection show significant performance gains on GPQA, particularly Llama-3.1-8B-Instruct. Notably, Qwen-2.5-7B-LC achieves GPQA performance slightly surpassing Bespoke-7B, a simpler replication of R1. The models also demonstrate modest improvements on MMLU-Pro, suggesting that the reasoning strategies learned from our dataset transfer effectively to general reasoning domains.

5.3 Implication on RL

After imitation learning to develop the long-form CoT reasoning skills, we move on the next phase—RLVR with GRPO (Shao et al., 2024)—to validate whether our collection serves as a reliable starting point for reinforcement learning. Due to GPU resource constraints for long-sequence RL, we train Qwen-2.5-0.5B on the Long CoT Collection and leverage it as the starting point for

RL. Based on the NuminaMATH (Jia LI and Polu, 2024), we filter samples to include only those with integer answers, resulting in a set of 10K examples. The policies are trained with a 16K max token length, using 16 samples per example for GRPO. For verifiable rewards, following three types of reward functions are employed.

Reward Functions. There are three reward functions we employed, which are generally used for RL: (1) Length Reward: We use the function $1 - \left| \frac{\min(x,y)}{\max(x,y)} - 1 \right|$ that measures the difference between the length of sampled thought and o1-mini's thought on a scale of 0 to 1. (2) Answer Reward: An outcome-based reward following Yeo et al. (2025). Specifically, we parse the answer span and compare it with the answer using latex2sympy, (3) Format Reward: We check whether the model responses include the parable answer span.

Results. Figure 1 represents the impact of our Long CoT Collection on the next RL phase. On both MATH500 and GPQA, the model initialized by training on our collection (i.e., Qwen-2.5-0.5B-LC) achieves 2-3x greater performance gains through RLVR compared to the base model (i.e., Qwen-2.5-0.5B), effectively mitigating the cold start problem. This indicates that the Long CoT Collection serves as a reliable starting point for RL, showing the potential to enable more stable learning even under sparse reward signals and finally leading to greater performance gains.

6 Thought Budget Control

One of the major issues with long-sequence reasoning models is overthinking—generating an unnecessarily large number of tokens for simple problems. For instance, QwQ-32B produces around 1,500 tokens for a basic question like '1+1+3?'. Similarly, OpenAI's O-series models offer three types—low, medium, high—based on computa-
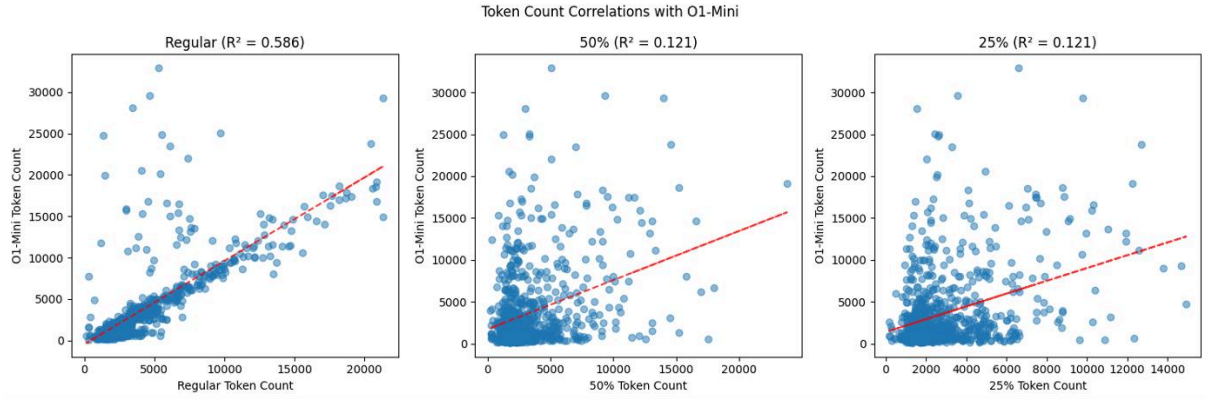
Figure 7: The Pearson correlation (R^2) between generated tokens and o1-mini thought tokens. We leverage 100% budget (*left*), 50% budget (*mid*), and 25% budget (*right*) to generate the collections of long CoT rationales.
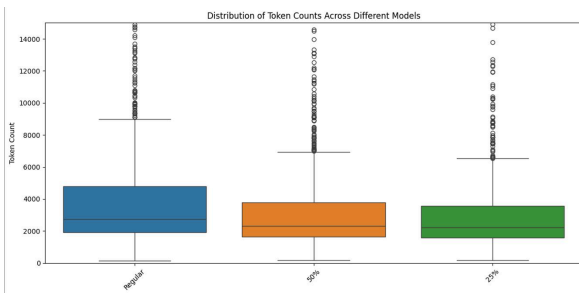

Figure 8: The token count distribution for each collection generated by adjusting the thought budget.

Data Used	100%	50%	25%
MATH500	66.6	60.7	57.6

Table 2: The results of policies on MATH500, which trained on each Long CoT Collection.

tional budget, allowing users to adjust the thinking budget according to the task complexity. Now, having control on the thought budget is crucial for effectively managing the problem of overthinking.

6.1 Controlling the Thought Budget During the Data Collection Process

As described in Section 3.2.2, when synthesizing long rationles from short CoT LLMs, our collection process first generates reasoning outlines by estimating the number of outlines needed for each instance and then producing a sequence of reasoning outlines accordingly. This allows us to control the length of the generated rationales by enforcing the number of outlines needed. We finally craft three versions of the Long CoT Collection by additionally constructing two more sets, each constrained to use only 25% and 50% of the original budget.

6.2 Analysis on the Budget-Controlled Collection

Figure 7 illustrates the correlation (R^2) between the tokens of the generated rationales and o1-mini thought tokens. It demonstrates that as we reduce the thought budget and generate relatively shorter rationales, the correlation with o1-mini thought to-

kens weakens. Moreover, we figure out that excessively reducing the thought budget—specifically to 25%—disrupts rationale generation by forcing too much information into too few reasoning outlines, making the reasoning more confusing.

We also investigate the distribution of each collection (i.e., 100%, 50%, and 25%). As presented in Figure 8, a reduction in the thought budget results in a corresponding decrease in the average token length of the collection. Furthermore, policies trained with access to larger budgets exhibit superior reasoning capability compared to those trained under more constrained budgets (Table 2).

7 Conclusion

This paper investigates the feasibility of generating long CoT datasets using LLMs trained on short CoT rationales. We present a pipeline for building the Long CoT Collection using short CoT LLMs, where the collection process offers controllability over the thought budget. This gives us the ability to regulate the length of the generated rationales and provides a way to address overthinking—one of the major challenges faced by LRMs. While training on our dataset did not lead to dramatic improvements over direct distillation from LRMs, our extensive experiments show that once moving into the RL phase, policies initialized with our dataset achieved 2-3x greater performance gains compared to those without it. This highlights the strength of our dataset as an reliable foundation for RL.

Limitations and Future Work

Application on Expert Domains. An exciting next step is to apply our pipeline to expert domains. While our dataset has proven to be a reliable starting point for RL in math and general reasoning tasks, we anticipate its potential to generalize further across a wide range of specialized domains.

Scaling Up to Larger Models. Although we employ 7B-8B models during phase 1 learning (i.e., supervised fine-tuning), we use a 0.5B model for phase 2 (i.e., reinforcement learning) since the largest model that fits within our GPU resources (16 A100 40GB GPUs) is 0.5B parameters.

Using Diverse Teacher LRMs. We only consider o1 for the reference LRM used in our dataset construction process. While we choose o1 due to its representativeness, our approach can be further applied to other LRMs that partially disclose their reasoning processes.

Acknowledgements

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev,

Arthur Hinsvark, Arun Rao, Aston Zhang, and 3 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2022. Unsolved problems in ml safety. *Preprint*, arXiv:2109.13916.

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

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? *Preprint*, arXiv:2411.16489.

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

Bespoke Labs. 2025. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. Accessed: 2025-01-22.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Zhenwen Liang, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. 2024. Improving llm reasoning through scaling inference computation with collaborative verification. *arXiv preprint arXiv:2410.05318*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *Preprint*, arXiv:2310.01889.

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

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

Mathematical Association of America. 2024. Aime.

OpenAI. 2024. Learning to reason with llms.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Bo Pang, Hanze Dong, Jiacheng Xu, Silvio Savarese, Yingbo Zhou, and Caiming Xiong. 2025. Bolt: Bootstrap long chain-of-thought in language models without distillation. *Preprint*, arXiv:2502.03860.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 24 others. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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 qa benchmark. *Preprint*, arXiv:2311.12022.

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

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

NovaSky Team. 2025a. Sky-t1: Fully open-source reasoning model with o1-preview performance in $450 budget. Accessed: 2025-01-09.

Open Thoughts Team. 2025b. Open Thoughts.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, Zhijiang Guo, Yaodong Yang, Muhan Zhang, and Debing Zhang. 2025. Redstar: Does scaling long-cot data unlock better slow-reasoning systems? *Preprint*, arXiv:2501.11284.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*.

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

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

Hongzhou Yu, Tianhao Cheng, Ying Cheng, and Rui Feng. 2025. Finemedlm-o1: Enhancing the medical reasoning ability of llm from supervised fine-tuning to test-time training. *Preprint*, arXiv:2501.09213.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, and 1 others. 2024a. Llama-berry: Pairwise optimization for o1-like

olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

Yuxiang Zhang, Shangxi Wu, Yuqi Yang, Jiangming Shu, Jinlin Xiao, Chao Kong, and Jitao Sang. 2024b. o1-coder: an o1 replication for coding. *Preprint*, arXiv:2412.00154.

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

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. 2023. Starling-7b: Improving llm helpfulness & harmlessness with rlaif.

A Details of the Data Construction

A.1 Base Dataset

Xu et al. (2024) propose a set of synthetic instruction data, Magpie, which covers a wide range of domains. From Magpie, a dataset consisting of 150K of the longest examples from reasoning, math, and coding & debugging categories was also released. Our 1K seed datasets and 100K long CoT collection are stems from the Magpie-Reasoning-150K.[1] Each data point is annotated with multiple subcategories along with its main category.

A.2 Details of Demonstration Retrieval

We leverage the main category and subcategories annotated in the dataset to retrieve demonstrations. We calculate the domain matching score by assigning 1 point for matching main categories and 0.2 points for each matching subcategory. The final retrieval score is computed by multiplying the domain matching score with the thought budget score, prioritizing samples with similar thought budgets.

A.3 Statistics

Figure 9 shows the distributions of reasoning steps, rationale lengths, and differences between reference and generated thought tokens. Our dataset contains sufficiently long rationales, with up to 30 reasoning steps and 20K thought tokens. The comparison between generated and reference thought tokens reveals similar distributions, suggesting our approach may help prevent under- and over-thinking issues (Wang et al., 2025) and enable short CoT LLMs to produce long rationales.

---

[1] https://huggingface.co/datasets/Magpie-Align/Magpie-Reasoning-V1-150K
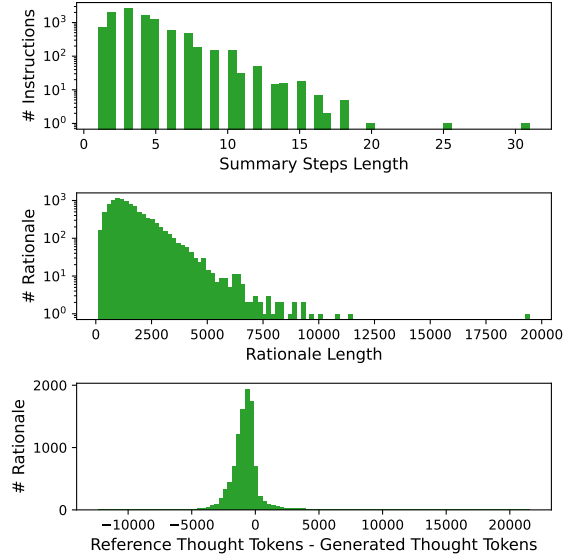
Figure 9: Distribution of the number of reasoning outlines (top), thought length (middle), and the difference between the length of the reference models' rationale and the generated rationale (bottom).

Phrase	GPT-4o	Deepseek-R1	Ours
"Let's"	37	92	100
"Wait"	0	100	4
"Okay"	0	100	47
"Verif-"	4	60	27
"?"	0	87	27
"!"	0	4	2

Table 3: Frequency of "Aha" moment phrases in CoT rationales across different methods, representing the proportion (%) of samples in which each phrase appears.

A.4 Analysis of Reasoning Triggers

Prior work reports that LRMs frequently use "Aha" moment phrases to explore better reasoning paths (Huang et al., 2024; DeepSeek-AI et al., 2025; Muennighoff et al., 2025a). These phrases serve not only as formatting elements but also as critical keywords that can steer the model's reasoning process, effectively guiding it towards more structured and thorough problem-solving. Thus, we check the frequency of these reasoning triggers, such as "Let's think," "Wait, I need to verify," and question marks indicating self-reflection. As shown in Table 3, while GPT-4o exhibits minimal use of these markers (primarily "Let's" at 37%), Deepseek-R1 employs them extensively across all categories.

B Implementation Details

B.1 Datasets

In Section C, LIMO (Ye et al., 2025) dataset serves as a test-bed to assess the potential of our initialized SFT model in RL. The LIMO dataset stems from NuminaMath-CoT, featuring meticulously annotated problems from high school to advanced competition levels, AIME, and MATH. It contains 817 meticulously selected math problems and solutions refined through human curation based on solutions generated by LRMs such as DeepSeek-R1. Most importantly, it includes only problems with verifiable answers, limited to integers within three digits.

B.2 Supervised Fine-tuning

We employ two base models: Qwen-2.5-7B-Instruct and Llama-3.1-8B-Instruct. The models are trained on the long CoT collection using 4 A100 GPUs. We adopt LLaMA-Factory (Zheng et al., 2024), a unified framework that integrates a suite of cutting-edge efficient training methods, to efficiently train the models.[2] Detailed hyperparameters used for the training are provided in Table 4.

B.3 RLVR

Due to the limited GPU budgets, we employ Qwen-2.5-0.5B as a base model for training on long sequences with GRPO. For our RL stage, we select a synthetic math dataset, NuminaMath (Jia LI and Polu, 2024), filtering problems based on Olympiads and AMC, resulting in a total of 10K problems. We adopt OpenRLHF (Hu et al., 2024), a framework designed to simplify and streamline RLHF training, and leverage RingAttnetion (Liu et al., 2023) to enable training on long sequences. Our RL stage is conducted on 16 A100 GPUs, and details about hyperparameters are in Table 5.

B.4 Benchmark Details

AIME 2024 contains 30 problems administered on January 31–February 1, 2024. AIME assesses mathematical problem-solving across various domains including arithmetic, algebra, counting, geometry, number theory, and probability. MATH (Hendrycks et al., 2021b) comprises competition mathematics problems spanning different difficulty levels. Following previous work by OpenAI (Lightman et al., 2023), we use the same sub-

Hyperparameters	Value
Base Model	Qwen-2.5-7B-Instruct / Llama-3.1-8B-Instruct
Torch dtype	BF16
Epoch	3
Train Data	Long CoT Collection
Learning Rate	5e-6
Max Seq. Length	8,192
Batch Size	1
Gradient Accumulation	8

Table 4: Hyperparameters used in the supervised fine-tuning.

Hyperparameters	Value
Base Model	Qwen-2.5-0.5B-LC
Torch dtype	BF16
Epoch	5
Train Data	NunimaMath-CoT
Learning Rate	5e-7
Max Seq. Length	16,384
Batch Size	64
Gradient Accumulation	1
Samples per Prompt	16

Table 5: Hyperparameters used for GRPO

set of 500 problems for evaluation. Along with the mathematical benchmarks, we test our model on the general reasoning benchmarks, GPQA Diamond (Rein et al., 2023), a dataset consists of 198 doctorate-level questions across Biology, Chemistry, and Physics, and MMLU-Pro (Wang et al., 2024b) an enhanced version of MMLU (Hendrycks et al., 2021a) with a stronger focus on reasoning capabilities.

B.5 Inference

all experiments are conducted with a temperature of 0.6 and a maximum token length of 16K, except for BoN sampling. For BoN, we use top-p decoding with $p = 0.95$ and $t = 1.0$. Each model generates n=1, 2, 4, 8, 16, and 32 responses on MATH-500 and AIME2024, and selects the one that contain correct answer. Since we focus on reasoning tasks, where correct answer is clearly defined, the results of BoN are equal to Pass@n. To efficiently test models across diverse benchmarks, we utilize Simple-Eval, an open-source library from OpenAI.[3]

---

[2]https://github.com/hiyouga/LLaMA-Factory

[3]https://github.com/openai/simple-evals
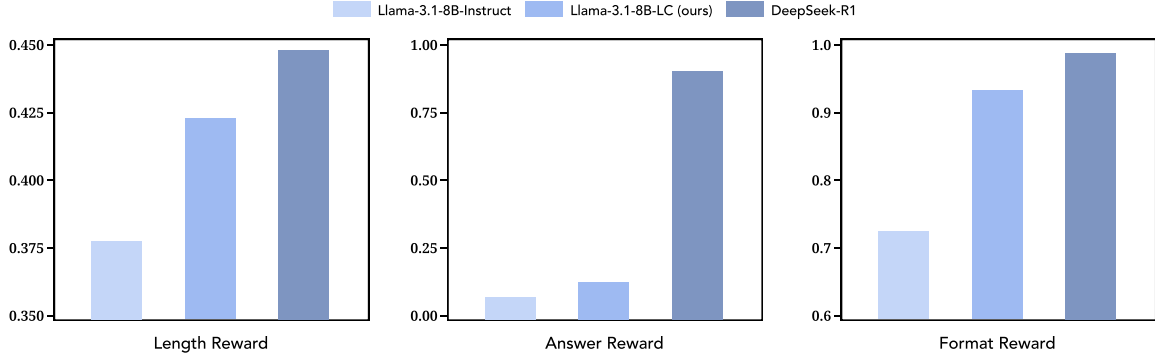
Figure 10: Length, answer, and format rewards across three models in LIMO dataset (Ye et al., 2025).

C Impact on the Verifiable Rewards

The success of RL is highly dependent on the SFT model (Ouyang et al., 2022). We investigate the effect of initializing the SFT model with our dataset to the rewards for RL. We utilize three reward functions aforementioned in Section 5.3.

Figure 10 compares the averaged rewards of our model, Llama-3.1-8B-LC, with the baselines, Llama-3.1-8B-Instruct and R1. Among the three models, our model shows the highest length reward, suggesting the effectiveness of our dataset construction process in efficiently allocating thought tokens. Furthermore, our model's higher answer reward compared to Llama-3.1-8B-Instruct indicates its potential as an effective starting point for RL.

D Details on Analyses

D.1 Comparison on Thought Budget

To compare thought budgets across different models, we employ model-specific token counting methods. For OpenAI's LRMs, we calculate the thought tokens by subtracting the response sequence tokens from the total completion tokens provided in the API response. For R1, which provides complete responses, we extract the content between `<think>` and `</think>` tags and count tokens using the GPT tokenizer. Similarly, for our model's responses, we measure the token count of sequences within the `<thought>` and `</thought>` tags.

D.2 Details of the CoT Quality Analyses

We use o3-mini as a judge and ask the model to identify which reasoning path is better based on the given criteria. The model chooses among the available options - A, B, or tie - where the two models' responses are randomly assigned to A and

B for unbiased comparison.

E Examples of the Long CoT Collection

We provide several examples from the Long CoT Collection:

- An example of our Long CoT Collection Figure 14

- An example response of Llama-3.1-8B-Instruct (Ours), which trained on the Long CoT Collection: Figure 15

F Prompts

These are the prompts we utilized in our study:

- Prompt for the step-wise long CoT generation: Figure 16

- Prompt for the correctness filtering: Figure 17

- Prompt for the CoT quality analyses in Section 4.1: Figure 18, 19, and 20.

G Usage of AI Assistant

We used ChatGPT for simple grammar correction and paraphrasing our draft.

H Artifact Licenses

- **magpie-reasoning-V1 dataset**: META LLAMA 3 COMMUNITY LICENSE AGREEMENT

- **AIME2024**: MIT license

- **MATH-500**: MIT license

- **LIMO dataset**: MIT license

- **GPQA diamond dataset**: cc-by-4.0

Figure 11: Results of Llama-3.1-8B-LC on MMLU-Pro broken down by domain.

Figure 13: Heatmap of the thought budget function, defined as $1 - \left| \frac{\min(x,y)}{\max(x,y)} - 1 \right|$, where x and y are positive integers. Brighter regions indicate higher rewards, which occur when x and y are closer in value.

Figure 12: Results of Qwen-2.5-7B-LC on MMLU-Pro broken down by domain.

❓ Question

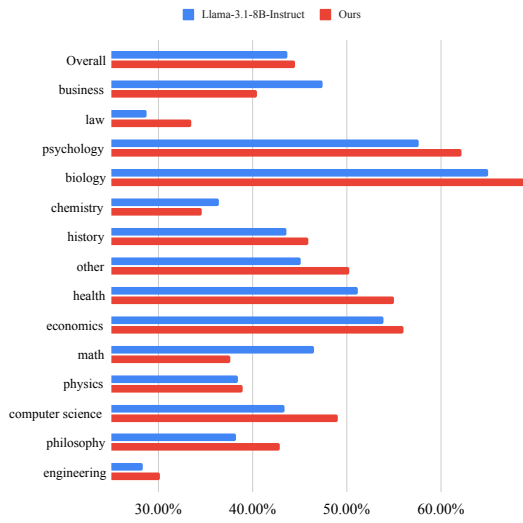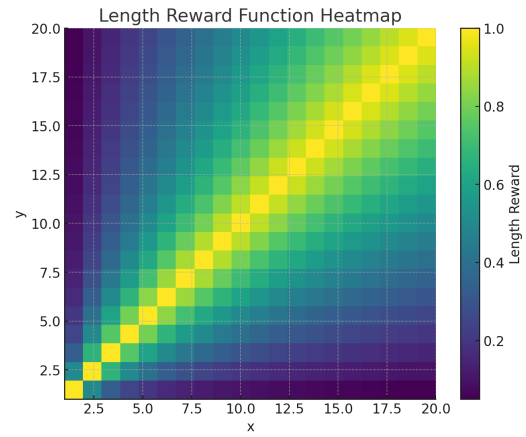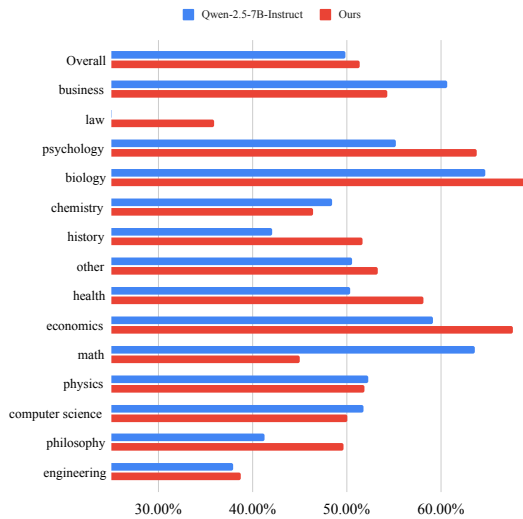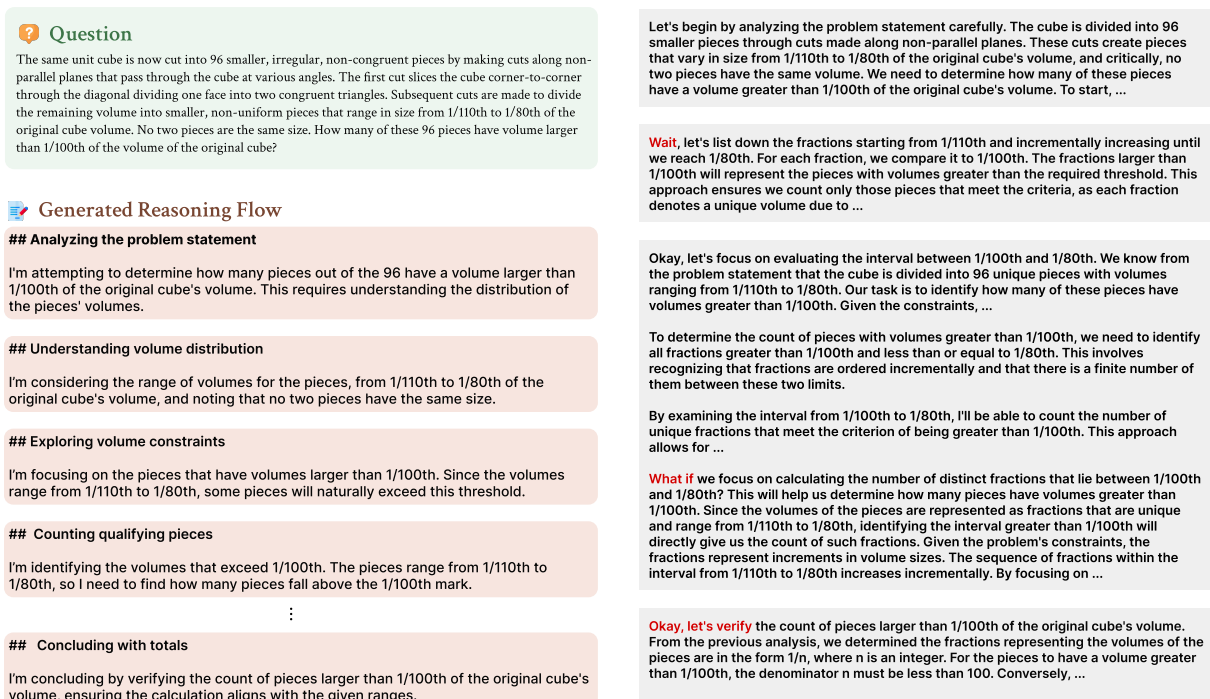The same unit cube is now cut into 96 smaller, irregular, non-congruent pieces by making cuts along non-parallel planes that pass through the cube at various angles. The first cut slices the cube corner-to-corner through the diagonal dividing one face into two congruent triangles. Subsequent cuts are made to divide the remaining volume into smaller, non-uniform pieces that range in size from 1/110th to 1/80th of the original cube volume. No two pieces are the same size. How many of these 96 pieces have volume larger than 1/100th of the volume of the original cube?

📝 Generated Reasoning Flow

Analyzing the problem statement

I'm attempting to determine how many pieces out of the 96 have a volume larger than 1/100th of the original cube's volume. This requires understanding the distribution of the pieces' volumes.

Understanding volume distribution

I'm considering the range of volumes for the pieces, from 1/110th to 1/80th of the original cube's volume, and noting that no two pieces have the same size.

Exploring volume constraints

I'm focusing on the pieces that have volumes larger than 1/100th. Since the volumes range from 1/110th to 1/80th, some pieces will naturally exceed this threshold.

Counting qualifying pieces

I'm identifying the volumes that exceed 1/100th. The pieces range from 1/110th to 1/80th, so I need to find how many pieces fall above the 1/100th mark.

⋮

Concluding with totals

I'm concluding by verifying the count of pieces larger than 1/100th of the original cube's volume, ensuring the calculation aligns with the given ranges.

💡 Generated Long CoT

Let's begin by analyzing the problem statement carefully. The cube is divided into 96 smaller pieces through cuts made along non-parallel planes. These cuts create pieces that vary in size from 1/110th to 1/80th of the original cube's volume, and critically, no two pieces have the same volume. We need to determine how many of these pieces have a volume greater than 1/100th of the original cube's volume. To start, ...

Wait, let's list down the fractions starting from 1/110th and incrementally increasing until we reach 1/80th. For each fraction, we compare it to 1/100th. The fractions larger than 1/100th will represent the pieces with volumes greater than the required threshold. This approach ensures we count only those pieces that meet the criteria, as each fraction denotes a unique volume due to ...

Okay, let's focus on evaluating the interval between 1/100th and 1/80th. We know from the problem statement that the cube is divided into 96 unique pieces with volumes ranging from 1/110th to 1/80th. Our task is to identify how many of these pieces have volumes greater than 1/100th. Given the constraints, ...

To determine the count of pieces with volumes greater than 1/100th, we need to identify all fractions greater than 1/100th and less than or equal to 1/80th. This involves recognizing that fractions are ordered incrementally and that there is a finite number of them between these two limits.

By examining the interval from 1/100th to 1/80th, I'll be able to count the number of unique fractions that meet the criterion of being greater than 1/100th. This approach allows for ...

What if we focus on calculating the number of distinct fractions that lie between 1/100th and 1/80th? This will help us determine how many pieces have volumes greater than 1/100th. Since the volumes of the pieces are represented as fractions that are unique and range from 1/110th to 1/80th, identifying the interval greater than 1/100th will directly give us the count of such fractions. Given the problem's constraints, the fractions represent increments in volume sizes. The sequence of fractions within the interval from 1/110th to 1/80th increases incrementally. By focusing on ...

Okay, let's verify the count of pieces larger than 1/100th of the original cube's volume. From the previous analysis, we determined the fractions representing the volumes of the pieces are in the form 1/n, where n is an integer. For the pieces to have a volume greater than 1/100th, the denominator n must be less than 100. Conversely, ...

Figure 14: An example instance from the Long CoT Collection.

❓ Question

Suppose that the measurement of time during the day is converted to the metric system so that each day has 10 metric hours, and each metric hour has 100 metric minutes. Digital clocks would then be produced that would read 9:99 just before midnight, 0:00 at midnight, 1:25 at the former 3:00 AM, and 7:50 at the former 6:00 PM. After the conversion, a person who wanted to wake up at the equivalent of the former 6:36 AM would set his new digital alarm clock for A:BC , where A , B , and C are digits. Find $100\text{A}+10\text{B}+\text{C}$.

💡 Generated Long CoT

Let's begin by understanding the conversion from standard time to the metric time system. We know that in the standard system, the day is divided into 24 hours, with each hour consisting of 60 minutes. In contrast, the metric system divides the day into 10 metric hours, with each metric hour containing 100 metric minutes.

To solve the problem, we need to convert a standard time of 6:36 AM into the metric time system. This involves finding the equivalent metric hours and minutes for 6:36 AM.

1. **Calculate the proportion of the day**: First, consider that the total time from midnight to 6:36 AM is a fraction of the entire 24-hour day. We can express this as a ratio: ...

What if we examine the calculation of the equivalent metric minutes more closely to ensure the conversion is accurate?

1. **Reassessing the Conversion**: We've already converted the standard time of 6:36 AM into the metric system, where each metric hour comprises 100 metric minutes. In our previous steps, we determined that 36 standard minutes equate to 60 metric minutes. Let's confirm this conversion step-by-step.

2. **Fraction of the Metric Hour**: The 36 standard minutes must be converted to the metric system, where a full metric hour consists of 100 metric minutes. We previously calculated the equivalent fraction of the metric hour as:
$$\frac{36}{60} \times 100 = 60 \text{ metric minutes}$$
This calculation is consistent with the relationship between standard and metric time, confirming that 36 minutes indeed translates to 60 metric minutes.

3. **Verification within Metric Constraints**: ...

With this verification, we ensure the metric conversion aligns with the problem's requirements. This consistency supports the conversion of the standard time into the metric system accurately, as needed for further calculations. Thus, the calculated time remains 2:60 in the metric system, and we can confidently proceed with the solution.

Okay, since we have confirmed that the equivalent metric time for 6:36 AM is 2:60, let's proceed to calculate the expression $(100\text{A} + 10\text{B} + \text{C})$.

1. **Identify the Values of A, B, and C**: ...

...

3. **Conclusion**:
   - The calculation confirms that the value of the expression is 260. This reflects the correct conversion and representation of the time according to the problem's requirements.

Figure 15: An example response from Llama-3.1-8B-LC (Ours), which trained on the Long CoT Collection.

Prompt for step-wise long CoT generation

Generate a step-by-step solution with a detailed rationale for solving the question according to the given reasoning strategy. ...

Question: {question}

Previous rationales:
{previous_reasoning}

Current step reasoning strategy:
{current_strategy}

Next step reasoning strategy:
{next_strategy}

Reasoning for the current step:

Figure 16: Prompt used for stepwise long CoT generation.

Prompt for correctness filtering

You are an AI assistant for grading a science problem. The user will provide you with the question itself, an attempt made by a student and the correct answer to the problem. Your job is to judge whether the attempt is correct by comparing it with the correct answer. If the expected solution concludes with a number or choice, there should be no ambiguity. If the expected solution involves going through the entire reasoning process, you should judge the attempt based on whether the reasoning process is correct with correct answer if helpful.
The user will provide the attempt and the correct answer in the following format:

Problem
{problem}

Attempt
{attempt}

Correct answer
{solution}

Explain your reasoning, and end your response on a new line with only "Yes" or "No" (without quotes).

Figure 17: Prompt used for filtering the incorrect rationales.

```
Prompt for qualitative analysis:  Reasoning Flow

Which rationale uses better reasoning strategies? Don't simply judge based on the length.
Choose between three options A, B, TIE. Only output 'A','B', 'TIE' without any explanation.

Question:
{question}

A:
{A}

B:
{B}
```

Figure 18: Prompt used for qualitative analysis on Reasoning Flow. We assign the position of A/B randomly.

```
Prompt for qualitative analysis:  Reasoning Strategy

Which rationale uses better reasoning flow? Don't simply judge based on the length.
Choose between three options A, B, TIE. Only output 'A','B', 'TIE' without any explanation.

Question:
{question}

A:
{A}

B:
{B}
```

Figure 19: Prompt used for qualitative analysis on Reasoning Strategy. We assign the position of A/B randomly.

```
Prompt for qualitative analysis:  Correctness

Which rationale is more correct? Choose between two options A and B. Only output 'A','B',
'TIE' without any explanation.

Question:
{question}

A:
{A}

B:
{B}
```

Figure 20: Prompt used for qualitative analysis on Correctness. We assign the position of A/B randomly.