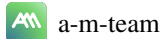# How Difficulty-Aware Staged Reinforcement Learning Enhances LLMs' Reasoning Capabilities: A Preliminary Experimental Study

Yunjie Ji,   Sitong Zhao,   Xiaoyu Tian,   Haotian Wang,
Shuaiting Chen,   Yiping Peng,   Han Zhao,   Xiangang Li

a-m-team

Abstract

Enhancing the reasoning capabilities of Large Language Models (LLMs) with efficiency and scalability remains a fundamental challenge in artificial intelligence research. This paper presents a rigorous experimental investigation into how difficulty-aware staged reinforcement learning (RL) strategies can substantially improve LLM reasoning performance. Through systematic analysis, we demonstrate that strategically selecting training data according to well-defined difficulty levels markedly enhances RL optimization. Moreover, we introduce a staged training methodology, progressively exposing models to increasingly challenging tasks, further amplifying reasoning capabilities. Our findings reveal significant cross-domain benefits when simultaneously training models on mathematical reasoning and code generation tasks. Notably, our proposed approach enables a 1.5B parameter model to achieve an accuracy of 42.3% on the AIME-2024 benchmark, 89.5% on the MATH-500 benchmark. These results underscore the efficacy of our method in advancing the reasoning proficiency of LLMs. We will open-source our datasets on GitHub[1] and Hugging Face[2].

1   Introduction

Enhancing the reasoning capabilities of Large Language Models (LLMs) while maintaining scalability and efficiency remains a key challenge in modern AI research. While contemporary LLMs, such as GPT-4 [11], Claude 3 [2], and DeepSeek-R1 [17], have demonstrated impressive performance across various tasks, further enhancing their reasoning abilities in specialized domains like mathematical reasoning and code generation is still an ongoing challenge [22, 15]. Recently, several promising strategies have emerged, including sophisticated prompting techniques like Chain-of-Thought (CoT) [21] and Test-Time Scaling methods [20], alongside reinforcement learning (RL)-based approaches designed to optimize reasoning capabilities directly [12, 16].

Test-time scaling strategies have shown notable success in improving reasoning during inference by using extended prompts or iterative processes, without retraining the model parameters. Techniques like self-consistency voting, extended chain-of-thought prompts, and Tree-of-Thought search [22, 26, 19] help models explore multiple potential solutions and reflect upon reasoning paths, thereby reducing logical errors and enhancing overall accuracy. However, these methods primarily build upon the model's existing capabilities, and do not address the underlying optimization of reasoning strategies.

---

[1] https://github.com/a-m-team/a-m-models
[2] https://huggingface.co/a-m-team

Reinforcement learning, on the other hand, offers a more direct approach to improving reasoning performance by explicitly rewarding desired reasoning outcomes. Models like DeepSeek-R1 have demonstrated the potential of RL to significantly improve performance on tasks such as mathematical problem-solving and code generation [17, 1]. However, applying RL to optimize reasoning performance comes with its own set of challenges, including the selection of suitable training data, managing the length of reasoning chains, and the quality of the initial baseline model.

In this paper, we focus on addressing these challenges through a systematic investigation of the factors influencing reinforcement learning-based reasoning optimization. Our experiments reveal that the careful selection of appropriately difficult training data, combined with a staged RL training approach, can significantly improve model performance. Specifically, we find that carefully selecting training data with appropriate difficulty levels, combined with a staged training approach, significantly improves model performance. This strategy leads to a performance improvement of 5.6% on the MATH-500 benchmark and 13.4% on the AIME-2024 benchmark compared to the base model.

Additionally, we demonstrate the benefits of mixing mathematical reasoning and code generation tasks within the same training process. By training the model on both types of tasks simultaneously, we observe improvements across both domains. Specifically, the performance on the AIME-2024 benchmark improved by 7.8%, the MATH-500 benchmark improved by 5.2%, and LiveCodeBench saw a 3.5% increase, compared to the base model.

Our contributions in this paper are as follows:

- We provide a detailed experimental analysis of how training data difficulty and staged training impact reinforcement learning optimization for reasoning tasks.

- We demonstrate the effectiveness of mixed-domain training, specifically combining mathematical reasoning and code generation, leading to significant performance improvements across both domains.

- We empirically validate our findings by showing a 13.4% improvement on AIME-2024 and a 5.5% improvement on MATH-500.

Our findings offer practical guidance for applying reinforcement learning to enhance reasoning capabilities in LLMs and contribute to the ongoing research on scaling models for complex, multi-domain reasoning tasks.

2 Related work

Since the release of DeepSeek R1[17], a growing number of research groups and open-source communities have attempted to replicate its remarkable reasoning capabilities. These replication efforts vary in scale and methodology, but collectively provide a valuable understanding of R1's training pipeline and reasoning behavior. In this section, we review key replication efforts and related studies, focusing on whether they provide publicly accessible reports or technical documentation, as well as how their outcomes compare with the original R1 model.

Open-R1[6] by Hugging Face is one of the most comprehensive and transparent replication initiatives. The project openly shares its implementation and datasets through GitHub and aims to fully reproduce the three-stage training pipeline of DeepSeek-R1: (1) R1-Distill for supervised fine-tuning via distilled reasoning data, (2) R1-Zero for zero-SFT reinforcement learning using self-generated rewards, and (3) the full multi-stage training involving SFT and RL. Open-R1 maintains detailed project documentation, datasets such as OpenR1-Math-220k, and benchmark results that show competitive performance compared to R1.

OpenThinker-32B[18], developed by researchers from Stanford and UC Berkeley, demonstrates that with only 114k carefully filtered reasoning samples, it is possible to fine-tune Qwen2.5-32B to match or even surpass the performance of DeepSeek-R1's 32B distilled model. Notably, OpenThinker-32B outperforms R1 on benchmarks such as MATH500 while using just 14% of the training data.

The SimpleRL-Reason[24] project from HKUST investigates the capability of small models trained with reinforcement learning to replicate R1-Zero-style behavior. Without any supervised pretraining, the authors apply PPO to fine-tune Qwen2.5-Math-7B using only 8K math problems. The resulting

model shows strong performance across various math reasoning tasks, outperforming baseline supervised fine-tuned models.

TinyZero[13], from UC Berkeley, focuses on low-cost replication of R1-Zero-like abilities using a lightweight reinforcement learning setup. The authors train a 3B model with less than $30 in compute budget, and demonstrate the emergence of self-corrective reasoning behavior on arithmetic and logic puzzles.

The Simple-GRPO[8] project from Fudan University proposes an extremely lightweight implementation of DeepSeek's Group Relative Policy Optimization (GRPO) algorithm. With only 300 lines of core code and training conducted on minimal hardware, the project demonstrates that "Aha moments" in R1-Zero-style training can be replicated at extremely low cost.

Datawhale-R1[3], developed by the Datawhale community, offers a Chinese-language tutorial and implementation for replicating R1-Zero behavior. While primarily intended for educational purposes, the project documents the full training process on a local Qwen-7B model, and reports trends consistent with those observed in larger replication efforts.

LIMO (Less Is More for Reasoning)[23] proposes an alternative to large-scale replication by demonstrating that only 817 high-quality samples are sufficient to fine-tune a Qwen-7B model to state-of-the-art performance across many math reasoning benchmarks. Their paper reports results such as 94.8% accuracy on MATH500 and 57.1% on AIME24, significantly outperforming previous models.

Dr. GRPO [9] analyzes the design and biases of the R1-Zero training framework. The authors identify issues such as length bias in GRPO optimization and propose Dr. GRPO, a debiased version of the algorithm. Their 7B-scale model achieves 43.3% on AIME24, setting a new SOTA among open-source R1-Zero-style replications.

DeepScaleR[10] introduces a novel approach to scaling reinforcement learning by fine-tuning a 1.5B model with 40,000 math problems, achieving state-of-the-art performance. Their model surpasses OpenAI's O1-Preview with a 43.1% Pass@1 accuracy on AIME2024, demonstrating significant improvements over previous models. The authors show that efficient scaling can be achieved by reducing computational costs while maintaining high performance, thanks to a distilled model and iterative context lengthening

In summary, these replication projects have collectively shed light on the reproducibility, efficiency, and limitations of R1-style training. By providing open-source code, high-quality datasets, and technical documentation, they offer valuable resources for future work on reasoning-capable language models.

3 Data Selection

3.1 Data Sources

The training data used in our reinforcement learning is primarily drawn from high-quality, math- and code-centric datasets such as `numina-math`[7] and `PRIME`[4]. We follow a meticulous data curation process, inspired by methodologies described in prior work, please refer to [25] for more details.

3.2 Difficulty Scoring

To assess the intrinsic difficulty of each data sample, we evaluate the performance of multiple reasoning models on the math and code problems. Specifically, we run `DeepSeek-R1-Distill-Qwen-1.5B`, `DeepSeek-R1-Distill-Qwen-7B`, `DeepSeek-R1-Distill-Qwen-32B` and `DeepSeek-R1` across the datasets multiple times. Each model's *pass rate* on individual problems is recorded and used as an approximate measure of difficulty.

To enhance reliability, we use a verifier to check each answer. For math problems, we first extract the answer part from the model's response and use rules to parse the content within the `\boxed{}` expressions. We then compare this extracted content with the ground truth answer and use the Hugging Face Math Verify tool[5] to check whether the two match. Any format inconsistencies or incorrect answers are assigned a score of zero.

$$\text{score\_math} = \begin{cases} 0, & \text{if model's response format is incorrect,} \\ 0, & \text{if model's answer is incorrect,} \\ 1, & \text{if model's answer is correct.} \end{cases}$$

For programming problems, we execute the model-generated code within a secure sandboxed environment. The outcome is quantified using the ratio of test cases passed to the total number of test cases:

$$\text{score\_code} = \frac{\texttt{\#passed\_test\_case}}{\texttt{\#test\_cases}}$$

Importantly, we discard any samples where the `DeepSeek-R1` model achieves a pass rate of zero, as such instances are either too difficult or unverifiable due to limitations of the verifier.

3.2.1 Data Filtering Strategy

Our data selection process is based on three distinct difficulty levels, which are designed to conduct comprehensive experiments.

Difficulty Level 1: This subset includes data samples where `DeepSeek-R1-Distill-Qwen-1.5B` achieves a pass rate strictly between 0 and 1 (i.e., pass rate $\in (0, 1)$). These samples are considered moderately challenging, as the model is partially successful, making them valuable for training on problems that are neither trivial nor unsolvable.

Difficulty Level 2: This level includes three type of samples:

- Samples where `DeepSeek-R1-Distill-Qwen-1.5B` fails completely (pass rate = 0), but `DeepSeek-R1-Distill-Qwen-7B` succeeds fully (pass rate = 1). This highlights the advantages of scaling up model size.
- Samples where `DeepSeek-R1-Distill-Qwen-1.5B` fails (pass rate = 0), but `DeepSeek-R1-Distill-Qwen-7B` partially succeeds (pass rate $\in (0, 1)$). These samples emphasize the robustness of larger models when smaller models fail.
- Samples where both `DeepSeek-R1-Distill-Qwen-1.5B` and `DeepSeek-R1-Distill-Qwen-7B` achieve partial success, useful for analyzing common challenges and exploring potential improvements.

Difficulty Level 3: This subset consists of difficult tasks where the model is less likely to succeed. These samples are chosen based on the performance of the `DeepSeek-R1-Distill-Qwen-32B` model:

- We retain all samples that the 32B model fails to solve, representing the hardest tasks in our dataset.
- For the math subset, we retain 50% of the samples successfully solved by the 32B model. For code tasks, we retain only 10% of such samples. This selective retention ensures that only the most challenging or borderline cases are kept for training.

This tiered approach ensures that the resulting dataset includes a balanced distribution of tasks across different difficulty levels, which facilitates subsequent experiments, such as evaluating model performance under varying training data difficulties and conducting staged experiments that gradually increase the difficulty of the training data.

4 Experiments

4.1 Experimental Setup

In our experiments, we utilized the GRPO (Group Relative Policy Optimization) [14] algorithm. The learning rate was set to 1×10^{-6}. For each prompt, 16 samples were generated, with the maximum

sequence length set to 16k tokens. The training batch size was configured to 128, and on-policy model parameter updates were performed throughout the training process. To control the balance between the policy's exploration and exploitation, we set the KL divergence coefficient to 0.001. We also incorporated an entropy loss term, with the entropy coefficient set to 0.001 to encourage a sufficiently high level of exploration during training. The initial policy model used in our experiments was `DeepSeek-R1-Distill-Qwen-1.5B`. For evaluation, we set the generation sequence length limit to 32k tokens, with the temperature set to 0.6 and top_p set to 0.95. We computed the Pass@1 accuracy as the evaluation metric. For AIME-2024 and MATH-500, we ran the evaluation 16 times, while for LiveCodeBench, we ran it 4 times.

4.2 Impact of Data Difficulty on Model Performance

In reinforcement learning (RL), an essential question is what type of data can facilitate better policy learning. To address this, we selected data based on difficulty level as a key criterion. The difficulty of the data was determined by the pass rate of different models, which served as a metric to categorize the data into three difficulty levels, as described in 3.2.1.

The initial rewards on the training set for the models were as follows: 0.58, 0.50, and 0.17. These results align with the difficulty trends that were set when constructing the three datasets. The performance of the model on datasets of varying difficulty levels is summarized below.
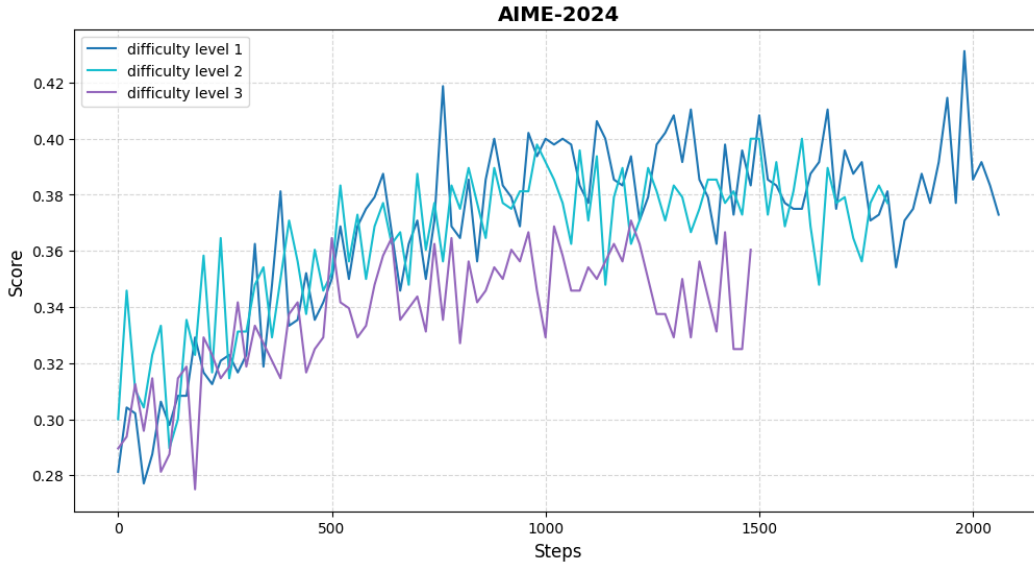

Figure 1: Performance of the model on the AIME-2024 benchmark during training with three different difficulty levels.

Figure 1 illustrates the performance trajectories of the model across these three difficulty levels. The results highlight the importance of selecting appropriately challenging RL training data. Based on our experiments, we found that data with an initial policy pass rate within the range (0, 1)—corresponding to difficulty level 1—yielded the best training results. This dataset offers a moderate challenge for the initial policy. This balance enables the model to generate both correct and incorrect responses during multiple rollouts for the same prompt, thus enhancing the effectiveness of RL training. In contrast, difficulty level 3, which represents highly challenging data, results in some instances where the model continuously fails to generate correct answers, which in turn reduces the model's convergence efficiency and overall performance.

In summary, carefully selecting RL training data based on appropriate difficulty metrics is critical. A moderate difficulty level enhances learning efficiency, balancing the need for adequate challenge against the risk of overwhelming the learning process with overly difficult scenarios.
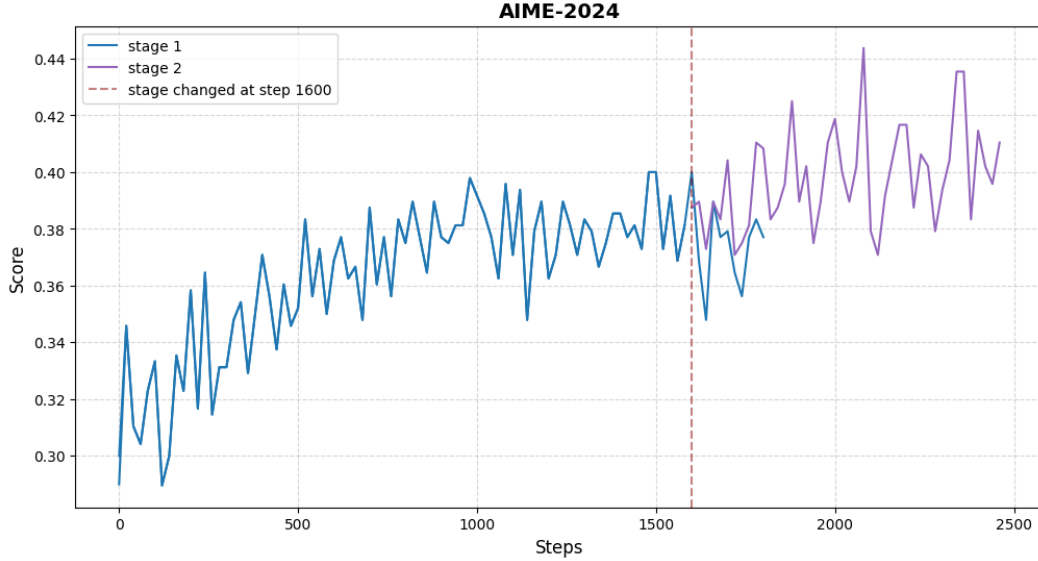
Figure 2: Performance of the model during staged RL training. The plot shows the model's score over time, with stage 1 (blue) and stage 2 (orange). The vertical dashed line indicates the point (step 1600) where the model transitioned from stage 1 to stage 2, resulting in performance improvements.

4.3 Staged RL Training

There are several reasons why we may wish to adopt a staged approach in reinforcement learning (RL) training, such as reducing training costs or further enhancing the model's performance. After conducting a series of experiments, we implemented the following staged RL training strategy.

In the first stage, we trained the model using data from difficulty level 2, following the training configurations outlined in 4.1. Once the model's performance plateaued, we transitioned to the second stage of training. During this phase, we chose to use difficulty level 3 data. Additionally, we adjusted the maximum sequence length for the rollouts to 24k.

Increasing the maximum sequence length led to a rapid increase in the length of the responses generated by the policy. However, a portion of these responses were truncated due to the length limitation. Providing negative feedback for these truncated responses is not ideal, as it could result in unfairly penalizing the model. Therefore, we opted not to compute the loss for these truncated samples. Furthermore, we decided to exclude the entropy loss from this phase of training.

When comparing this staged approach to the continuous training in the first stage, we observed that switching to the second stage resulted in sustained performance improvements, demonstrating the effectiveness of this strategy in further enhancing model capabilities.

4.4 Simultaneous Training on Mathematics and Code

To improve the model's capabilities in various domains, such as mathematics, programming, and general problem-solving, it is essential to include diverse types of prompts in the training dataset, along with corresponding reward rules or reward models. The distribution and difficulty of these different data types significantly impact the model's performance.

In our preliminary experiments, we explored training the model on both mathematical and programming tasks simultaneously. Our findings suggest that simply mixing these two types of training data can lead to improvements in both mathematical and programming proficiency. Specifically, we selected difficulty level 1 data for mathematics and difficulty level 3 data for coding, with a ratio of approximately 2.1:1. The mixing strategy employed involved incorporating both mathematical and coding prompts within the same batch during training.
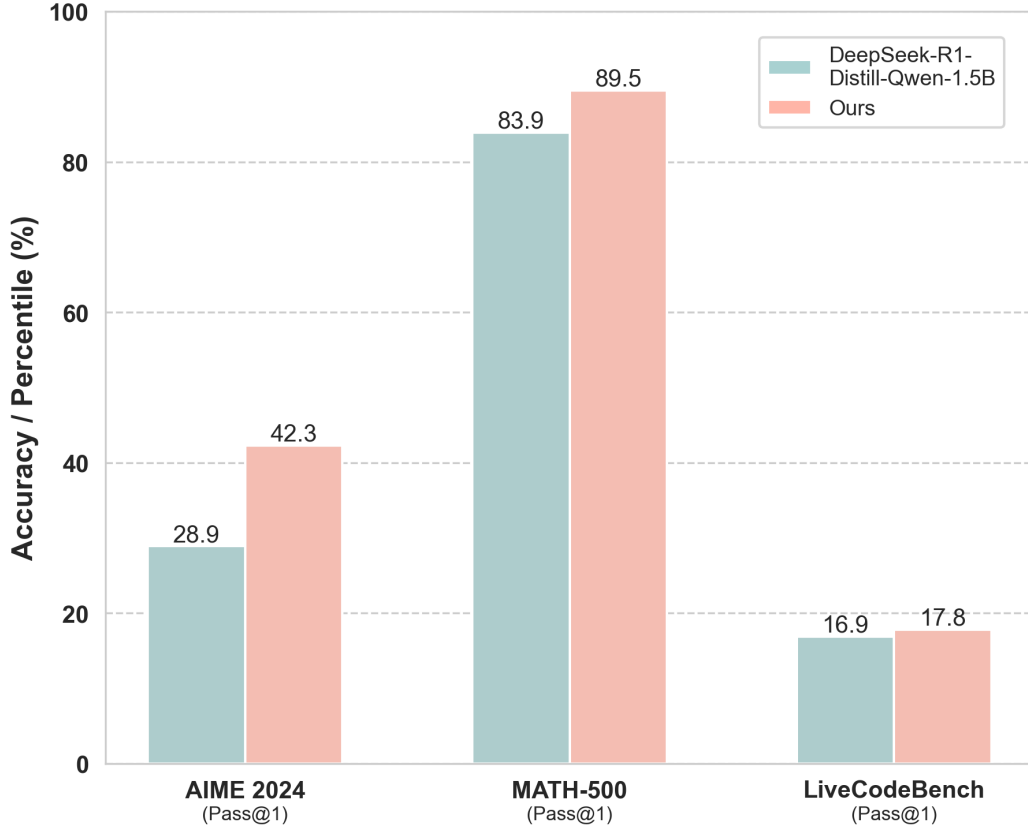
Figure 3: Our model, trained using a two-stage reinforcement learning (RL) approach, shows significant performance improvements on two mathematics-related benchmarks, AIME-2024 and MATH-500. However, due to the absence of code-related training data, its performance on Live-CodeBench is essentially the same as that of the base model.

Figure 4 presents the experimental results, demonstrating the effectiveness of this approach. Based on the conclusions drawn from 4.2, we hypothesize that incorporating difficulty level 1 coding data into the training mix would yield even more favorable outcomes, as it strikes a balance between challenge and learning efficiency.

5 Conclusion and Limitations

In this paper, we have explored the potential of using reinforcement learning (RL) to enhance the reasoning capabilities of Large Language Models (LLMs), specifically focusing on the careful selection of training data and staged training strategies. Our experiments demonstrated that by selecting appropriately challenging data and incorporating staged training, we can significantly improve the performance of LLMs on reasoning tasks. Notably, the performance gains were substantial on benchmarks such as MATH-500 and AIME-2024, with improvements of 5.6% and 13.4%, respectively. Furthermore, we showed that mixing mathematical reasoning and code generation tasks during training results in cross-domain improvements, providing strong evidence for the benefits of multi-domain training.

These findings provide practical guidance for future work on applying reinforcement learning to optimize reasoning in LLMs and highlight promising avenues for scaling models to handle more complex, multi-domain tasks.

However, despite the promising results, there are some limitations to our approach. Firstly, the selection of appropriate difficulty levels for training data is inherently dependent on the performance
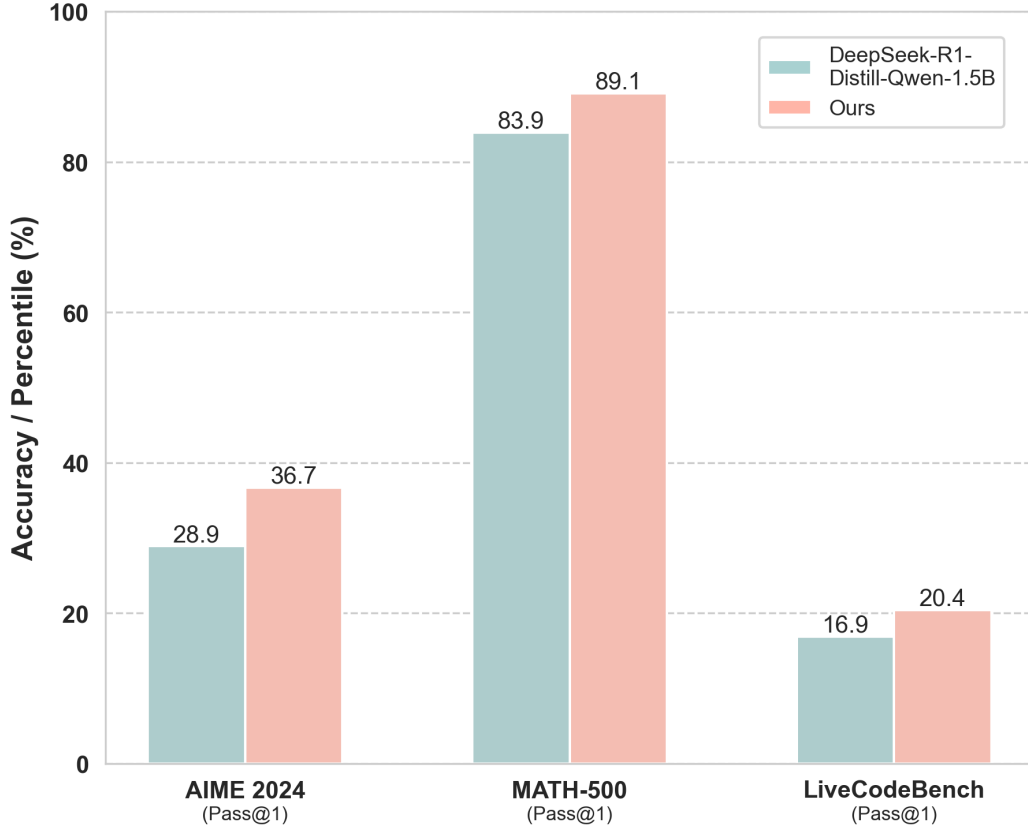
Figure 4: Performance comparison of models across different benchmarks: AIME-2024, MATH-500, and LiveCodeBench. Our model is simultaneously trained on math and code data.

of the initial models, which is costly. Further research is needed to refine these difficulty scoring methods and adapt them to a broader range of tasks. Secondly, while the staged RL training approach showed improvements, it is important to note that the performance gains were not uniform across all domains. While our staged approach reduced training costs, it is still computationally intensive, and further optimization could help reduce the resource requirements.

Lastly, the reliance on the quality of data and the underlying models poses another challenge. Ensuring the quality of training data, especially when dealing with complex reasoning tasks, remains a critical factor. Future work could focus on automating the data curation process and improving the robustness of the RL training pipeline to better handle noisy or ambiguous data.

References

[1] Kimi AI. Kimi-k1.5: Reinforcement learning enhanced llm reasoning. `https://github.com/Kimi-AI/Kimi-K1.5`, 2024. Accessed: March 2025.

[2] Anthropic. Claude 3: Advanced reasoning and alignment. `https://www.anthropic.com/claude-3`, 2024. Accessed: March 2025.

[3] Datawhale Community and SLRLab. Datawhale-r1: A chinese tutorial-level reproduction of deepseek-r1 zero, 2025. URL `https://github.com/datawhalechina/unlock-deepseek`.

[4] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[5] Hugging Face. Math-verify, 2025. URL `https://github.com/huggingface/Math-Verify`. Accessed: 2025-03-31.

[6] Hugging Face. Open-r1: A fully open reproduction of deepseek-r1, 2025. URL `https://huggingface.co/blog/open-r1`.

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

[8] Jiaqing Liang, Jinyi Han, Xinyi Wang, Zishang Jiang, Chengyuan Xiong, Boyu Zhu, Jie Shi, Weijia Li, Tingyun Li, and Yanghua Xiao. Kw-r1: A simple implementation of the grpo algorithm. `https://github.com/lsdefine/simple_GRPO`, 2025.

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

[10] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. `https://github.com/agentica-project/deepscaler`, 2025. Notion Blog.

[11] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL `https://arxiv.org/abs/2303.08774`.

[12] L. Ouyang et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL `https://arxiv.org/abs/2203.02155`.

[13] Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

[14] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[15] N. Shinn et al. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL `https://arxiv.org/abs/2303.11366`.

[16] D. Silver et al. Group relative policy optimization for language model training. Deep-Mind Technical Report, 2023. URL `https://deepmind.com/research/publications/group-relative-policy-optimization`. Accessed: March 2025.

[17] DeepSeek Team. Deepseek-r1: Reinforcement learning for advanced reasoning. `https://github.com/deepseek-ai/DeepSeek-RL`, 2024. Accessed: March 2025.

[18] OpenThoughts Team. Open Thoughts. https://open-thoughts.ai, January 2025.

[19] Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yunjie Ji, Yiping Peng, Han Zhao, and Xiangang Li. Think twice: Enhancing llm reasoning by scaling multi-round test-time thinking, 2025. URL `https://arxiv.org/abs/2503.19855`.

[20] X. Wang et al. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2023. URL `https://arxiv.org/abs/2203.11171`.

[21] J. Wei et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. URL `https://arxiv.org/abs/2201.11903`.

[22] S. Yao et al. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023. URL `https://arxiv.org/abs/2305.10601`.

[23] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. 2025.

[24] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL `https://arxiv.org/abs/2503.18892`.

[25] Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xiangang Li. 1.4 million open-source distilled reasoning dataset to empower large language model training, 2025. URL `https://arxiv.org/abs/2503.19633`.

[26] D. Zhou et al. Think twice: An inference-time approach to improve reasoning in llms. *arXiv preprint arXiv:2310.08387*, 2023. URL `https://arxiv.org/abs/2310.08387`.