demonstrate the effectiveness of the distillation technique, leaving the exploration of the RL stage to the broader research community. For details on distillation training, please see Appendix B.4.3.

Table 15 | Comparison of DeepSeek-R1 distilled models and other comparable models on reasoning-related benchmarks. Numbers in bold denote the performance is statistically significant (t−test with $p < 0.01$).

Model	AIME 2024		MATH	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	**72.6**	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70.0	**86.7**	**94.5**	**65.2**	**57.5**	**1633**

We evaluate the distilled models on AIME, GPQA, Codeforces, as well as MATH-500 (Lightman et al., 2024) and LiveCodeBench (Jain et al., 2024). For comparison, we use two well-established LLMs as baselines: GPT-4o and Claude-3.5-Sonnet. As shown in Table 15, the straightforward distillation of outputs from DeepSeek-R1 allows the distilled model, DeepSeek-R1-Distill-Qwen-1.5B, to surpass non-reasoning baselines on mathematical benchmarks. Notably, it is remarkable that a model with only 1.5 billion parameters achieves superior performance compared to the best closed-source models. Furthermore, model performance improves progressively as the parameter size of the student model increases.

Our experimental results demonstrate that smaller models can achieve strong performance through distillation. Furthermore, as shown in Appendix F, the distillation approach yields superior performance compared to reinforcement learning alone when applied to smaller model architectures. This finding has significant implications for democratizing AI access, as reduced computational requirements enable broader societal benefits.

F.1. Distillation v.s. Reinforcement Learning

Table 16 | Comparison of distilled and RL Models on Reasoning-Related Benchmarks.

Model	AIME 2024		MATH	GPQA Diamond	LiveCode Bench
	pass@1	cons@64	pass@1	pass@1	pass@1
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9
Qwen2.5-32B-Zero	47.0	60.0	91.6	55.0	40.2
DeepSeek-R1-Distill-Qwen-32B	**72.6**	**83.3**	**94.3**	**62.1**	**57.2**

In Section F, we can see that by distilling DeepSeek-R1, the small model can achieve impressive results. However, there is still one question left: can the model achieve comparable

Table 17 | Performance of different models on AIME 2024 and AIME 2025.

Average Score	**AIME 2024**	**AIME 2025**
GPT-4o-0513	9.3%	-
Qwen2-Math-7B-Instruct	7.9%	4.6%
Qwen2-Math-7B-Zero	22.3%	18.1%

performance through the large-scale RL training discussed in the paper without distillation?

To answer this question, we conduct large-scale RL training on Qwen2.5-32B-Base using math, code, and STEM data, training for over 10K steps, resulting in Qwen2.5-32B-Zero, as described in B.4.1. The experimental results, shown in Table 16, demonstrate that the 32B base model, after large-scale RL training, achieves performance on par with QwQ-32B-Preview. However, DeepSeek-R1-Distill-Qwen-32B, which is distilled from DeepSeek-R1, performs significantly better than Qwen2.5-32B-Zero across all benchmarks.

Therefore, we can draw two conclusions: First, distilling more powerful models into smaller ones yields excellent results, whereas smaller models relying on the large-scale RL mentioned in this paper require enormous computational power and may not even achieve the performance of distillation. Second, while distillation strategies are both economical and effective, advancing beyond the boundaries of human intelligence may still require more powerful base models and larger-scale reinforcement learning.

Apart from the experiment based on Qwen-2.5-32B, we conducted experiments on Qwen2-Math-7B (released August 2024) prior to the launch of the first reasoning model, OpenAI-o1 (September 2024), to ensure the base model was not exposed to any reasoning trajectory data. We trained Qwen2-Math-7B-Zero with approximately 10,000 policy gradient update steps. As shown in Table 17, Qwen2-Math-7B-Zero significantly outperformed the non-reasoning models like Qwen2-Math-7B-Instruct and GPT-4o. These results further demonstrate that the model can autonomously develop advanced reasoning strategies through large-scale reinforcement learning.

G. Discussion

G.1. Key Findings

We highlight our key findings, which may facilitate the community in better reproducing our work.

The importance of base checkpoint: During the initial phase of our development, we experimented with smaller-scale models, specifically a 7B dense model and a 16B Mixture-of-Experts (MoE) model, as the foundational architectures for RL training. However, these configurations consistently failed to yield meaningful improvements when evaluated on the AIME benchmark, which we employed as the primary validation set. We observed that as response lengths increased, these smaller models exhibited a tendency toward repetition and were unable to effectively leverage long chains of thought (CoT) to improve reasoning accuracy.

To address these limitations, we transitioned to larger-scale models, including a 32B dense model (Qwen, 2024b), a 230B MoE model (DeepSeek-AI, 2024a), and a 671B MoE model (DeepSeek-AI, 2024b). With these more capable architectures, we finally observed substantial

performance gains attributable to pure RL training. These findings suggest that the effectiveness of reinforcement learning from base models is highly dependent on the underlying model capacity. We therefore recommend that future research in this area prioritize the use of sufficiently large and expressive models when aiming to validate the efficacy of RL from scratch.

The importance of verifiers: The effectiveness of DeepSeek-R1-Zero is highly contingent upon the reliability and fidelity of the reward signal used during training. To date, our investigations indicate that two approaches—rule-based reward models (RMs) and LLMs to assess an answer's correctness against a predefined ground-truth—serve as robust mechanisms for mitigating issues related to reward hacking. The LLM-based evaluation framework demonstrates particular effectiveness for tasks with well-defined, concise answers, such as single-sentence or phrase-level responses. However, this method exhibits limited generalizability to more complex tasks, including open-ended generation and long-form writing, where the notion of correctness is inherently more subjective and nuanced.

Iterative pipeline: We propose a multi-stage training pipeline comprising both SFT and RL stages. The RL component enables the model to explore and discover optimal reasoning trajectories for tasks capabilities that cannot be fully realized through human-annotated reasoning traces alone. In particular, without the RL stage, long-chain reasoning patterns, such as those required in complex Chain-of-Thought (CoT) prompting, would remain largely unexplored. Conversely, the SFT stage plays a crucial role in tasks where reliable reward signals are difficult to define or model, such as open-ended question answering and creative writing. Therefore, both RL and SFT are indispensable components of our training pipeline. Exclusive reliance on RL can lead to reward hacking and suboptimal behavior in ill-posed tasks, while depending solely on SFT may prevent the model from optimizing its reasoning capabilities through exploration.

G.2. Unsuccessful Attempts

In the early stages of developing DeepSeek-R1, we also encountered failures and setbacks along the way. We share our failure experiences here to provide insights, but this does not imply that these approaches are incapable of developing effective reasoning models.

Process Reward Model (PRM) PRM is a reasonable method to guide the model toward better approaches for solving reasoning tasks (Lightman et al., 2024; Uesato et al., 2022; Wang et al., 2023a). However, in practice, PRM has three main limitations that may hinder its ultimate success. First, it is challenging to explicitly define a fine-grain step in general reasoning. Second, determining whether the current intermediate step is correct is a challenging task. Automated annotation using models may not yield satisfactory results, while manual annotation is not conducive to scaling up. Third, once a model-based PRM is introduced, it inevitably leads to reward hacking (Gao et al., 2022), and retraining the reward model needs additional training resources and it complicates the whole training pipeline. In conclusion, while PRM demonstrates a good ability to rerank the top-N responses generated by the model or assist in guided search (Snell et al., 2024), its advantages are limited compared to the additional computational overhead it introduces during the large-scale reinforcement learning process in our experiments.

Monte Carlo Tree Search (MCTS) Inspired by AlphaGo (Silver et al., 2017b) and AlphaZero (Silver et al., 2017a), we explored using Monte Carlo Tree Search (MCTS) to enhance test-time compute scalability. This approach involves breaking answers into smaller parts to allow the model to explore the solution space systematically. To facilitate this, we prompt the model to

generate multiple tags that correspond to specific reasoning steps necessary for the search. For training, we first use collected prompts to find answers via MCTS guided by a pre-trained value model. Subsequently, we use the resulting question-answer pairs to train both the actor model and the value model, iteratively refining the process.

However, this approach encounters several challenges when scaling up the training. First, unlike chess, where the search space is relatively well-defined, token generation presents an exponentially larger search space. To address this, we set a maximum extension limit for each node, but this can lead to the model getting stuck in local optima. Second, the value model directly influences the quality of generation since it guides each step of the search process. Training a fine-grained value model is inherently difficult, which makes it challenging for the model to iteratively improve. While AlphaGo's core success relied on training a value model to progressively enhance its performance, this principle proves difficult to replicate in our setup due to the complexities of token generation.

In conclusion, while MCTS can improve performance during inference when paired with a pre-trained value model, iteratively boosting model performance through self-search remains a significant challenge.

H. Related Work

H.1. Chain-of-thought Reasoning

Chain-of-thought (CoT) reasoning (Wei et al., 2022b) revolutionized how LLMs approach complex reasoning tasks by prompting them to generate intermediate reasoning steps before producing a final answer. This method significantly improved performance on benchmarks involving arithmetic, commonsense, and symbolic reasoning. Subsequent work explored its scope: Suzgun et al. (2023) demonstrated that CoT's effectiveness scales with model size, while Kojima et al. (2022) extended it to zero-shot settings by simply instructing models to "think step by step."

Building on CoT's framework, numerous "prompt engineering" techniques have been proposed to enhance model performance. Wang et al. (2023b) introduced self-consistency, a method that aggregates answers from multiple reasoning paths to improve robustness and accuracy. Zhou et al. (2023a) developed least-to-most prompting, which decomposes complex problems into sequential subquestions that are solved incrementally. Yao et al. (2023a) proposed tree-of-thoughts, enabling models to explore multiple reasoning branches simultaneously and perform deliberate decision-making through looking ahead or backtracking. Collectively, these approaches leverage human prior knowledge and more structured reasoning frameworks to enhance the reasoning capabilities of LLMs.

H.2. Scaling Inference-time Compute

As unsupervised pre-training scaling might be constrained by the amount of available human data (Kaplan et al., 2020; Muennighoff et al., 2023), scaling compute during inference has become even more critical (Snell et al., 2025). Broadly, we define methods that improve model performance by increasing inference compute as forms of scaling inference-time compute.

A straightforward approach trades compute for performance by generating multiple diverse reasoning chains and selecting the best answer. The optimal answer can be identified using a separate reranker (Brown et al., 2024; Cobbe et al., 2021), process-based reward models (Lightman et al., 2024; Uesato et al., 2022), or simply by selecting the most common answer

(Wang et al., 2023b). Search methods, such as Monte Carlo Tree Search and Beam Search, also guide exploration of the solution space more effectively (Feng et al., 2024; Hao et al., 2023; Trinh et al., 2024; Xin et al., 2024). Beyond parallel generation, self-correct techniques prompt or train models to iteratively critique and refine their outputs (Kumar et al., 2024; Madaan et al., 2023; Welleck et al., 2023), often incorporating external feedback to enhance reliability (Gou et al., 2024a; Yao et al., 2023b). Additionally, some methods improve performance by integrating tool use during testing, which is particularly effective for knowledge-intensive (Nakano et al., 2021) and compute-intensive tasks (Chen et al., 2025; Gou et al., 2024b; Schick et al., 2023). Test-time training (TTT) further updates the model during inference to boost performance (Akyürek et al., 2024; Sun et al., 2020). There are also various other inference-time scaling approaches that—either implicitly (Geiping et al., 2025) or explicitly (Zelikman et al., 2024)—allocate more compute for each token.

In contrast, our work shows that LLMs can achieve scalable improvements through additional RL compute and increased test-time compute (i.e., more tokens). We integrate the benefits of scaling at test time into a broader framework that uses reinforcement learning to incentivize enhanced in-context search abilities.

H.3. Reinforcement Learning for Reasoning Enhancement

Reinforcement Learning plays a pivotal role in aligning LLMs with human preferences (Bai et al., 2022; Ouyang et al., 2022). Despite its importance, few studies have focused on using RL to enhance reasoning capabilities. Traditional RL pipelines begin with SFT on high-quality human demonstrations, which provides a strong initialization and prevents mode collapse. Following this, a reward model is trained on human preferences, and the language model is subsequently optimized using methods such as PPO (Schulman et al., 2017) or DPO (Rafailov et al., 2023). Although this method works well for alignment, it risks constraining models to emulate human reasoning patterns, potentially hindering the discovery of novel problem-solving strategies.

Methods like STaR iteratively boost performance by fine-tuning on the model's self-generated chain-of-thought that leads to correct final answers (Singh et al., 2024; Yuan et al., 2023; Zelikman et al., 2022). Recent studies have also investigated the use of process-based rewards that emphasize both the correctness of final answers and the soundness of the reasoning processes (Lightman et al., 2024; Shao et al., 2024; Wang et al., 2023a). Unlike these methods, our work applies outcome-based RL directly to base language models without an initial SFT phase. This design choice encourages the emergence of innovative and unconstrained reasoning strategies, enabling the model to develop diverse solutions beyond mere imitation of human examples. Our approach also inspired further exploration in subsequent research (Face, 2025; Liu et al., 2025; Pan et al., 2025).

I. Open Weights, Code, and Data

To promote the development of the open-source community and industry ecosystem, we have made the model weights of DeepSeek-R1 and DeepSeek-R1-Zero publicly available on HuggingFace. In addition, we release DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, DeepSeek-R1-Distill-Qwen-14B, DeepSeek-R1-Distill-Qwen-32B, DeepSeek-R1-Distill-Llama-8B, DeepSeek-R1-Distill-Llama-70B.

Furthermore, we have released the fundamental model inference code (`https://github.com/deepseek-ai/DeepSeek-V3`) and provided detailed usage guidelines (`https://github.com/deepseek-ai/DeepSeek-R1`) on GitHub.