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.

Here is an example of running the inference code to interact with DeepSeek-R1:

```
# Download the model weights from Hugging Face
huggingface-cli download deepseek-ai/DeepSeek-R1 --local-dir
/path/to/DeepSeek-R1

# Clone DeepSeek-V3 GitHub repository
git clone https://github.com/deepseek-ai/DeepSeek-V3.git

# Install necessary dependencies
cd DeepSeek-R1/inference
pip install -r requirements.txt

# Convert Hugging Face model weights to a specific format (for running
the model on 16 H800 GPUs)
python convert.py --hf-ckpt-path /path/to/DeepSeek-R1 --save-path
/path/to/DeepSeek-R1-Demo --n-experts 256 --model-parallel 16

# Run the model and interact with it
torchrun --nnodes 2 --nproc-per-node 8 --node-rank $RANK --master-addr
$MASTER_ADDR generate.py --ckpt-path /path/to/DeepSeek-R1-Demo --config
configs/config_671B.json --interactive --temperature 0.7
--max-new-tokens 8192
```

We also release SFT and RL data to the public at xxx. In the review process, we upload the data as an attachment.

J. Evaluation Prompts and Settings

Table 18 | MMLU assesses a model's factual and conceptual understanding across 57 tasks spanning STEM (science, technology, engineering, mathematics), humanities, social sciences, and professional fields (e.g., law, medicine). The benchmark is commonly used to evaluate a model's ability to perform general knowledge reasoning and multitask proficiency across a diverse range of subjects and tasks. Here is an example of MMLU.

PROMPT
Answer the following multiple choice question. The last line of your response should be of the following format: 'Answer: $LETTER' (without quotes) where LETTER is one of ABCD. Think step by step before answering.
Which tool technology is associated with Neandertals?

A. Aurignacian
B. Acheulean
C. Mousterian
D. both b and c

Evaluation
Parse the last line in response to judge if the choice equals to ground truth.