arXiv:2506.14731v2 [cs.CL] 18 Jun 2025

Ring-lite: Scalable Reasoning via C3PO-Stabilized Reinforcement Learning for LLMs

Ling Team, Inclusion AI[*]

[*]See Contributions section (Sec. 6) for full author list.

We present `Ring-lite`*, a Mixture-of-Experts (MoE)-based large language model optimized via reinforcement learning (RL) to achieve efficient and robust reasoning capabilities. Built upon the publicly available Ling-lite model, a 16.8 billion parameter model with 2.75 billion activated parameters, our approach matches the performance of state-of-the-art (SOTA) small-scale reasoning models on challenging benchmarks (e.g., AIME, LiveCodeBench, GPQA-Diamond) while activating only one-third of the parameters required by comparable models. To accomplish this, we introduce a joint training pipeline integrating distillation with RL, revealing undocumented challenges in MoE RL training. First, we identify optimization instability during RL training, and we propose Constrained Contextual Computation Policy Optimization(**C3PO**), a novel approach that enhances training stability and improves computational throughput via algorithm-system co-design methodology. Second, we empirically demonstrate that selecting distillation checkpoints based on **entropy loss** for RL training, rather than validation metrics, yields superior performance-efficiency trade-offs in subsequent RL training. Finally, we develop a **two-stage** training paradigm to harmonize multi-domain data integration, addressing domain conflicts that arise in training with mixed dataset. We will release the model, dataset, and code.*

Date: Jun 17, 2025
Code: `https://github.com/inclusionAI/Ring`

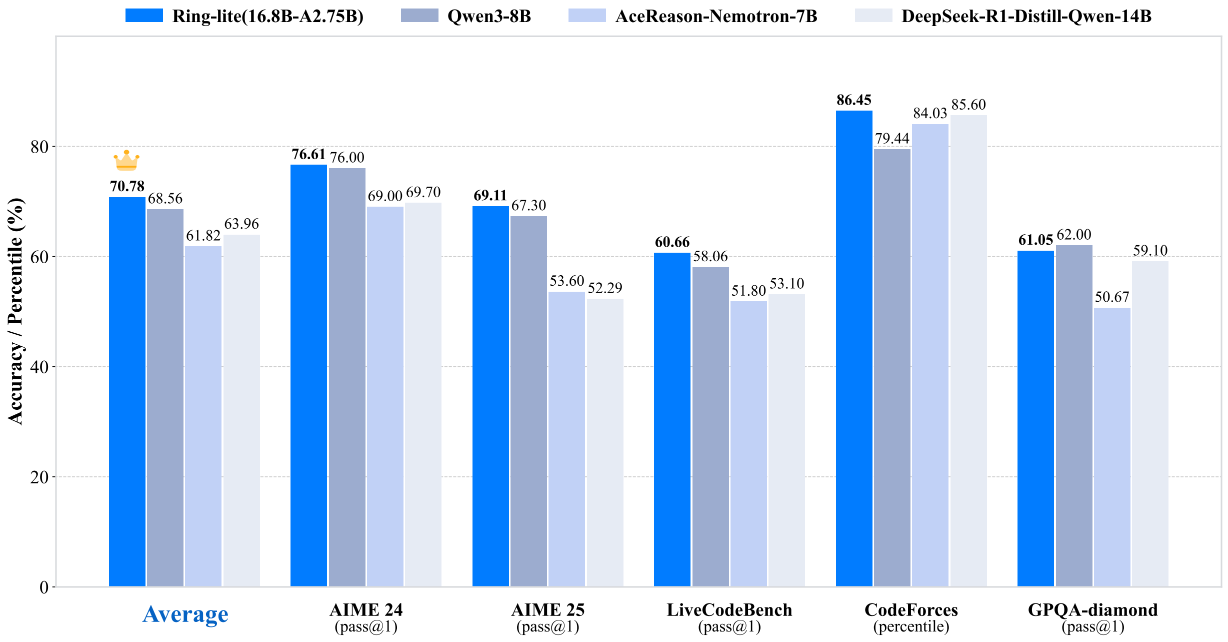


Figure 1 Benchmark performance of Ring-lite

1 Introduction

The remarkable success of the OpenAI-O1 series models (OpenAI, 2024) and DeepSeek-R1 (DeepSeek-AI, 2025) has demonstrated the substantial potential of large-scale reinforcement learning (RL) for complex reasoning tasks, attracting significant research attention. However, the detailed methodologies employed in the training of these models have not been fully disclosed, creating a significant knowledge gap that hinders further advancements in the field. This lack of transparency in training techniques and strategies impedes the reproduction and extension of these results by other researchers.

Recent advancements, such as DAPO (Yu et al., 2025), Open-Reasoner-Zero (Hu et al., 2025), and DeepCoder (Luo et al., 2025a), have demonstrated competitive reasoning performance through task-specific RL strategies, accompanied by publicly released models and datasets. However, their contributions remain narrowly scoped, focusing predominantly on isolated domains such as mathematics or code generation, with limited cross-task generalization. Furthermore, while current research has largely focused on dense model architectures (Yu et al., 2025; Zhang et al., 2025), scant attention has been devoted to exploring the potential of Mixture of Experts (MoE) (DeepSeek-AI, 2025; Ling-Team, 2025) paradigms in this context. Of particular concern is the persistent challenge of training stability—a fundamental prerequisite for scaling RL-based reasoning systems that remains systematically unaddressed. The dynamic interaction between specialized experts in MoE architectures introduces complex gradient synchronization and parameter update conflicts, often manifesting as oscillating loss or disaster forgetting of acquired reasoning skills. Without robust methodologies to ensure stable training in large-scale RL training, the theoretical advantages of MoE frameworks cannot be fully realized in practice.

In this work, we introduce `Ring-lite`, a fully open-source Mixture of Experts (MoE) reasoning model designed to enhance multi-domain reasoning capabilities built upon the publicly available Ling-lite model (Ling-Team, 2025). To the best of our knowledge, this is the first work that integrates an open training framework, open training data, and an open model, specifically targeting the domains of mathematics, coding, and STEM. Furthermore, `Ring-lite` systematically delves into the instability issues prevalent in RL training and the conflicts arising from capability integration across domains. To solve the instability problem, we propose Constrained Contextual Computation Policy Optimization(**C3PO**), a novel token-level optimization framework for reinforcement training. The experimental results obtained by our model on complex reasoning tasks, coupled with its more stable reward curves compared to the widely-used conventional Group Relative Policy Optimization (GRPO) method, substantiate the efficacy of our approach. Our model, `Ring-lite` with 16.8B total parameters and 2.75B activation parameters, establishes state-of-the-art performance across mathematical reasoning, code generation, and STEM problem-solving benchmarks, demonstrating superior performance by matching or surpassing dense models with under 10B parameters, the standard baseline for comparable architectures according to the scaling law analyses (Ling-Team, 2025). To our knowledge, `Ring-lite` represents the first publicly available mixture-of-experts (MoE) system operating at this parameter-efficiency frontier, serving both as a methodological innovation in efficient architectural design through dynamic sparse activation and as an open-access resource that lowers barriers to cutting-edge AI research. This dual contribution advances both theoretical understanding of neural scaling laws and practical democratization of high-performance language model technologies. Specifically, `Ring-lite` achieves impressive scores of 76.61% and 69.11% on AIME2024 and AIME2025, two challenging math competition-style benchmarks, 60.66% and 86.45% on LiveCodeBench and Codeforces, two challenging code contest benchmarks for code generation, 61.05% on GPQA-diamond, the graduate-Level science QA benchmark. It surpasses Qwen3-8B (Yang et al., 2025) in average and approaches the performance of other top-tier reasoning models.

In short, our work marks a pivotal advance in the democratization of AI research and development, as it provides the broader community with a fully open-source solution. By doing so, our work significantly lowers the barrier to entry for exploring multi-domain reasoning, empowering researchers and practitioners alike to contribute to and benefit from this burgeoning field. The main contributions are summarized as follows.

- For the first time, we open-source a multi-domain MoE reasoning model [1], encompassing a open source infrastructure (framework), training methodologies, and training datasets. The entire transparent training pipeline is detailed, including Long-CoT Supervised Fine-Tuning (SFT) and reasoning-specific reinforcement learning (RL).
- We identify a critical challenge in the training instability of reasoning models and propose C3PO, a framework that implements a fixed training token size (budget) to eliminate response length variance and select high-entropy base models to stabilize learning dynamics. The framework integrates a reinforcement learning methodology grounded in an algorithm-engineered co-design paradigm, thereby not only ensuring long-term training stability but also achieving significant gains in computational efficiency.
- Spanning multiple domains (math, code and science), we observed inter-domain data conflict and introduced a capability integration method (stage-wise training and balanced data mixing) to address this issue.

The structure of this work proceeds as follows: Section 2 details the dataset curation process, including data cleaning and filtering. Section 3 systematically outlines the methodological framework, emphasizing its contributions and implementation specifics. Finally, Section 4 evaluates the model's performance against established benchmarks and synthesizes critical insights gleaned from both quantitative results and qualitative observations.

2 Data

Our training dataset comprises two components: (1) long Chain-of-Thought (Long-CoT) supervised fine-tuning (SFT) data, employed to train a cold-start model, and (2) reinforcement learning (RL) data, designed to enhance reasoning capabilities.

2.1 Long-CoT Data

To activate a base model's reasoning capability, a comprehensive dataset of high-quality samples exhibiting Long-CoT reasoning patterns was curated. The query pool was sourced from open-source repositories and further enriched through synthetic generation using large language models (LLMs). To ensure the production of high-fidelity responses with Long-CoT, we implemented an iterative refinement pipeline that synergistically combines automated model generation, expert manual annotation, and rejection sampling mechanisms. After that, rigorous data-cleansing protocols were applied, including detection and removal of repetitive patterns, mixed-language artifacts, and other noise sources, to yield a robust and high-quality dataset. The final data is predominantly dominated by three major domains: Mathematics (64.5%), Code (25.5%), and Science (9.2%, encompassing

[1] https://github.com/inclusionAI/Ring

some high-quality and difficult samples generated by SHARP (Wu et al., 2025)). The remaining portion of the dataset includes contributions from other categories, such as medicine and history domains. In short, our long-CoT SFT dataset enabled effective multi-domain reasoning (spanning mathematics, coding, and science), providing a robust initialization for subsequent reinforcement learning training.

2.2 RL Training Data

2.2.1 Domain-Specific Reasoning Data

- **Math** We begin by sourcing a wide range of mathematical problems to enrich the diversity and coverage of our math reasoning dataset. These problems are mainly obtained from two channels: open-source datasets and self-collected data. For open-source datasets, we included datasets that have been meticulously curated for reinforcement learning, including BigMath (Albalak et al., 2025), DeepScaleR (Luo et al., 2025b), DAPO (Yu et al., 2025), DeepMath-103K (He et al., 2025b), etc. To further expand our dataset, we crawled online math forums and collected authentic school examinations. Specifically, we extracted problems from the contest section of the Art of Problem Solving (AoPS) website[2], which archives comprehensive records of historical mathematics competitions from diverse regions and educational levels. Additionally, we gathered a wide range of human-written problems utilized in school exams and mathematics competitions across various educational stages, such as high school and college. This extensive process yielded an initial collection of more than tens of thousands of math problems. We then applied stringent data cleansing and filtering protocols to ensure quality and relevance, ultimately refining the dataset to include over 73,000 high-quality math problems suitable for our reinforcement learning processes.

- **Code** The dataset was curated from open-source programming competition resources, primarily drawn from CodeContest (Li et al., 2022), TACO (Li et al., 2023) and APPS (Hendrycks et al., 2021), additionally some problems from the QOJ online judge platform [3]. To ensure data quality and training suitability, a multi-stage filtration process was implemented. First, test cases exhibiting format inconsistencies—such as erroneous line breaks or extraneous spaces—were systematically removed, along with truncated content marked by ellipses or incomplete patterns. Subsequently, all "Accepted"(AC) solutions underwent rigorous validation through our code sandbox environment. This verification step eliminated submissions with unresolved external dependencies and discarded implementations that failed extended test cases due to computational inefficiencies (*e.g.*, $O(n^2)$ algorithms for $n > 10^5$) or memory overflow vulnerabilities. Output standardization was enforced by normalizing whitespace conventions and aligning floating-point precision thresholds across platforms to mitigate inconsistencies in evaluation criteria. To ensure integrity, only problems accompanied by at least one fully validated AC solution capable of passing all associated test cases were retained, thereby preserving practical problems with existing solution. Semantic deduplication was applied to remove overlapping problems from public coding benchmarks, minimizing the risk of evaluation bias through contamination control. The final curated dataset comprises approximately 14,000 code samples, each accompanied by verified executable solutions and rigorously validated test cases, establishing a robust foundation for reward computation and model training.

[2]https://artofproblemsolving.com/community/c13_contest_collections
[3]https://qoj.ac

- **Science** For the science domain, our RL training data construction followed a three-stage evolution to ensure quality and difficulty alignment. Initially, we sourced open datasets such as Nemotron-CrossThink (Akter et al., 2025) and SCP-116K (Lu et al., 2025), etc., to establish a baseline for scientific reasoning. As model capabilities improved, we employed the SHARP (Wu et al., 2025) synthesis pipeline to generate harder, verifiable problems. However, due to the difficulty ceiling and verification limitations of synthetic and open-source data, our final RL training relied exclusively on a third-stage dataset. This consisted of a proprietary collection of high-difficulty, human-annotated science problems drawn from advanced natural science domains. Sources included Olympiad competitions and graduate-level (Master's and PhD) exams. We then applied a rigorous curation process—encompassing quality filtering, answer verification, and domain-specific tagging—resulting in a refined set of 3,833 high-quality scientific problems suitable for reinforcement learning.

2.3 Data Curation

The efficacy of the reinforcement learning process is heavily dependent on the quality of the training datasets. Through our initial investigations, we discover that data contamination issue persists even in the widely adopted open-source datasets. To ensure a high-quality training dataset for reinforcement learning, we developed an extensive and rigorous data curation pipeline, comprising several stages designed to ensure the complete decontamination of our data, thus making it readily prepared for RL training. The core components of our data processing protocol are primarily divided into the following three critical phases:

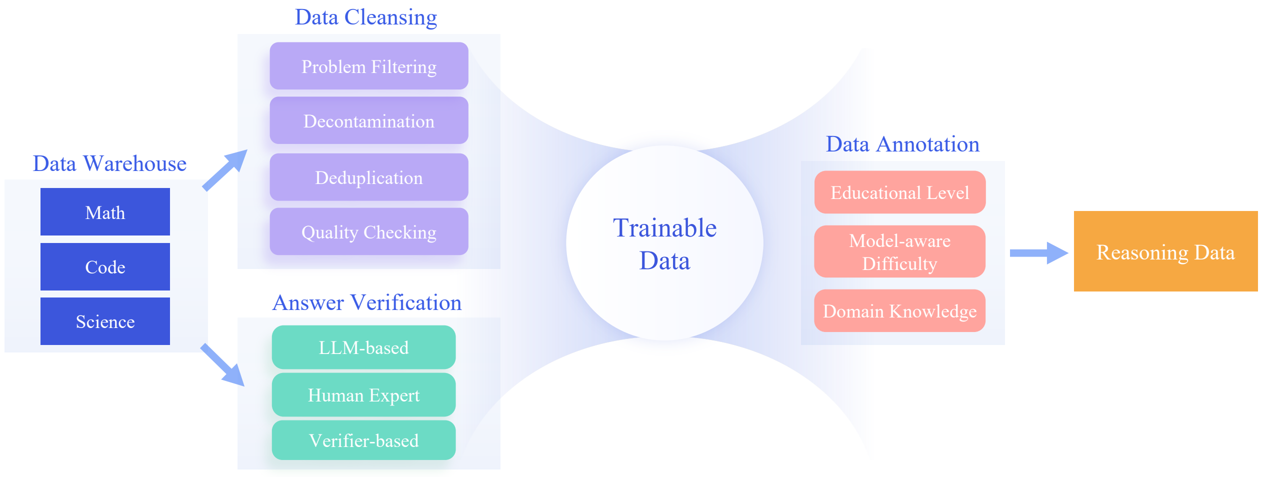


Figure 2 The data curation pipeline of `Ring-lite`

Data Cleansing We first exclude problems with invalid character, images, multi-subquestions, and those lacking valid answers. We conducted strict both character-based and semantic-based deduplication and decontamination on the dataset to ensure strict data cleansing. We also remove problems which cannot be uniquely sovled or susceptible to be easily guessed, such as multiple-choice questions, and problems that can be answered with True/False, Yes/No, etc.

Answer Verification To ensure the correctness of answers associated with problems in our dataset, we conduct thorough verification using diverse approaches. Specifically, we employ an LLM-based method to assess the quality of each answer. We utilize LLMs of different sizes to generate multiple individual solutions for each problem. Based on the verifiers used in RL training, we calculate the model-aware pass rate. Additionally, we engage human experts to manually annotate the answers. Problems that do not pass either verification method are excluded from our dataset.

Data Annotation To optimize the data selection strategy, we meticulously annotate each reasoning problem. Specifically, each problem is labeled with multi-dimensional attributes, such as data source, educational level, domain-specific knowledge, and more. For instance, we use the Mathematical Subject Classification (MSC) categories to assess the themes of our math problems. Additionally, we provide model-aware difficulty by computing the solve rate based on our distilled model. Problems that receive all correct solutions are deemed inefficient for RL training; therefore, we remove those problems. Conversely, problems that are unsolvable by both our distilled model and DeepSeek-R1 are also discarded to ensure that the remaining data contribute effectively to policy gradient updates in reinforcement learning.

3 Method

3.1 Preliminary

Group **R**elative **P**olicy **O**ptimization (**GRPO**) algorithm is widely used such as DeepSeek-R1, Qwen3 and so on. For each question-answer pair (q, a) in the training dataset $\mathcal{D}$, we generate K responses (i.i.d.) through the policy model $\pi_{\theta_{\text{old}}}$. The reward R_i of the response y_i is determined by the reward model or rule-based verifier. GRPO estimates the advantage via group-normalized rewards instead of the value model, $A_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^K)}{\text{std}(\{R_i\}_{i=1}^K)}$. Specifically, the GRPO loss is formulated as:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{(q,a)\sim\mathcal{D},\{y_i\}_{i=1}^K\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{K}\sum_{i=1}^{K}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\left(\min\left(r_{i,t}(\theta)A_{i,t},\ \text{clip}\Big(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\Big)A_{i,t} \right) - \beta D_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) \right) \right], \tag{1}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|q, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|q, y_{i,<t})}$ and ε is the clip bound. $D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})$ is the token-level KL loss, keeping the policy model π_θ not far from the reference policy π_{ref}.

3.2 C3PO: Constrained Contextual Computation Policy Optimization

Observation. While long-CoT reasoning remains essential for handling complex tasks, current GRPO methods demonstrate systemic length-related issues that fundamentally undermine training stability, especially for the long-CoT reasoning models. Specifically, our analysis reveals two following critical issues affecting the training stability:

- **Within-step length bias:** Within a single batch, unpacked responses of varying lengths induce substantial gradient bias under the GRPO objective, as initially identified in pioneering studies (Yu et al., 2025; Liu et al., 2025). This bias primarily arises from the length-normalized gradient estimation by normalizing per-response rewards through division by their token counts (termed per-token scaling), the procedure systematically amplifies gradient magnitudes for shorter sequences while attenuating them for longer ones. While recent research (Yu et al., 2025; Liu et al., 2025) has introduced token-level loss mechanisms to address within-step length bias, the persistence of across-step gradient variance continues to pose challenges, as elaborated below.

- **Across-step gradient variance:** During RL training with exploratory mechanisms, the policy model's generated responses exhibit substantial stochastic variance in sequence length. This dynamic sequence length variation induces non-trivial optimization challenges for token-level optimizers, as fluctuating lengths create training token inconsistencies that propagate through the learning pipeline. As empirically validated in Figure 7, highly-variation length fluctuation (Figure 7a) results in an abnormal gradient characteristics (Figure 7b), ultimately lead to premature reward collapse (Figure 7c).

In addition to the challenges associated with training instability, our empirical observations revealed that variations in response length significantly influence training efficiency. Specifically, longer response sequences result in increased inference and training latency, whereas shorter sequences compromise computational throughput efficiency, as shown in Figure 7d.

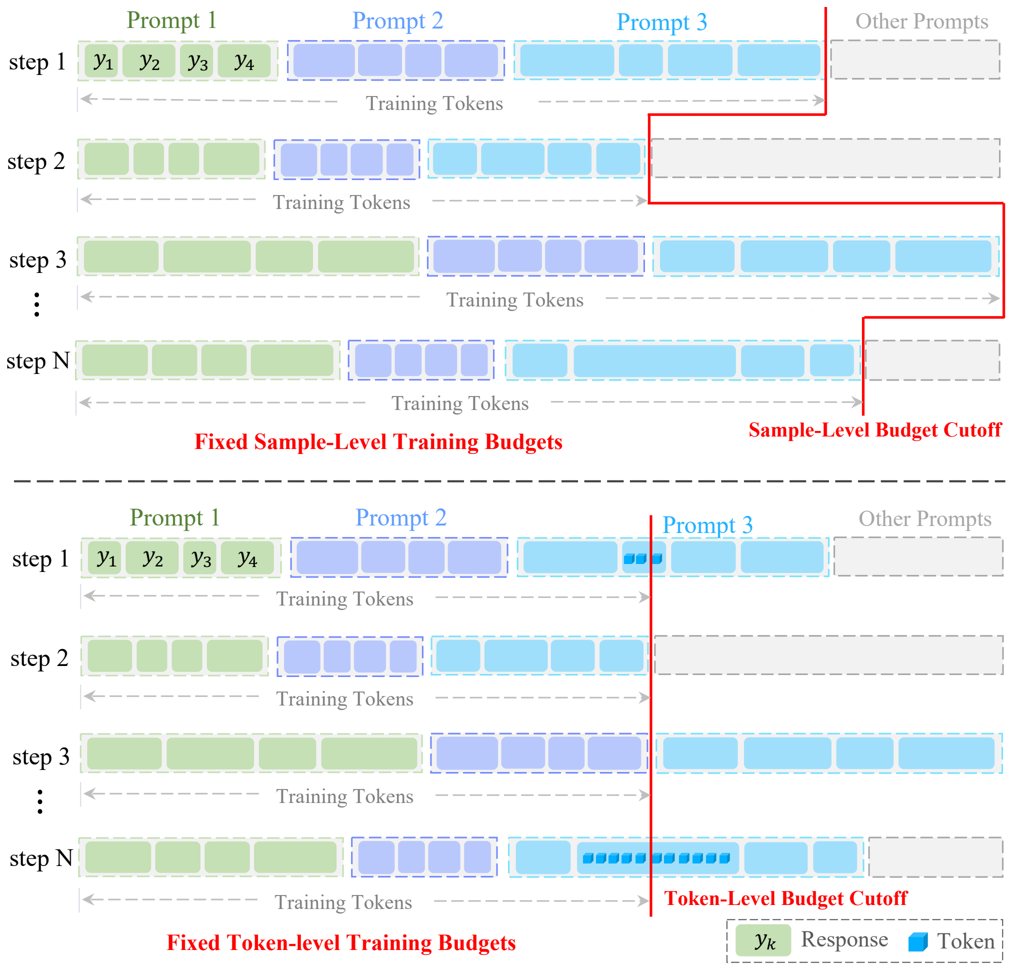

Prompt 1
Prompt 2
Prompt 3
Other Prompts
step 1
y_1 y_2 y_3 y_4
Training Tokens
step 2
step 3
step N
Fixed Sample-Level Training Budgets
Sample-Level Budget Cutoff
Token-Level Budget Cutoff
Fixed Token-level Training Budgets
y_k Response
Token

Figure 3 The comparison between our C3PO strategy and the widely-used dynamic sampling strategy, The C3PO performs token truncation when the token count exceeds the budget, after advantage computation but prior to gradient backpropagation

Methodology. Building upon these empirical observations, we posit that synergistic algorithm-engineering co-design constitutes a foundational requirement for achieving stable and scalable reinforcement learning training. To translate this principle into practice, we introduce Constrained Contextual Computation Policy Optimization(**C3PO**), an innovative token-level optimization framework designed to mitigate training instability while enhancing throughput consistency. The core innovation lies in establishing a formalized computational budgeting system that imposes explicit constraints on gradient contributions at the token level, thus ensuring homogeneous gradient contributions across variable-length sequences. With the training token budget, the GRPO loss function in Eq. 1 is reformulated as follows,

$$\begin{aligned}\mathcal{L}_{\text{C3PO}}(\theta) = -\mathbb{E}_{\{q,a\}_{l=1}^{L}\sim\mathcal{D},\{y_i\}_{i=1}^{K}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \\ \left[\frac{1}{\Phi}\sum_{i=1}^{|\mathcal{S}|}\sum_{t=1}^{|y_i|}\mathbb{I}\left[y_{i,t}\in\Psi\right]\left(\min\left(r_{i,t}(\theta)A_{i,t},\ \text{clip}\left(r_{i,t}(\theta),1-\varepsilon,1+\varepsilon\right)A_{i,t}\right)-\beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right)\right], \\ \text{s.t.}\quad |\Psi| = \Phi\end{aligned} \tag{2}$$

where Φ is the training token budget (i.e., constrained contextual computation), Ψ is the selected tokens by custom sampling strategy. $\mathcal{S}$ is the selected responses for training, L is the query size per step and K denotes the group size, $|y_i|$ denotes the token size for $i-$th responses. In our experiment the set Ψ is sampled as below:

$$\Psi = \{(y_1, y_2, \cdots, y_N)|y_i \in \mathcal{B}\},\quad \text{s.t. } N \leq |\mathcal{B}|, \sum_{j=1}^{N-1}|y_j| < \Phi, \sum_{j=1}^{N}|y_j| \geq \Phi \tag{3}$$

where $\mathcal{B} = \{(q,a);\{y_i\}_{i=1}^{K}\}_{l=1}^{L}$ is the entire set in a training step. During a practical training step, we implement token-budgeted dynamic sampling in the training phase. For each batch, we employ a greedy algorithm to iteratively select sufficient responses $\mathcal{S}$ whose cumulative token count closely exceeds or equals the predefined token budget Φ.

By implementing a fixed token budget per optimization step, C3PO systematically mitigates GRPO's sensitivity to variations in individual sequence lengths. This design facilitates two critical improvements: 1) Homogeneous Gradient Scaling: The uniform factor $1/\Phi$ ensures equivalent gradient contributions across responses of varying token lengths, resolving the disproportionate weighting bias between short and long sequences inherent in conventional approaches. Furthermore, such a design mitigates abnormal gradient magnitudes caused by fluctuations in response length, effectively preventing the destabilization of training dynamics and subsequent reward collapse; 2) Deterministic Training Dynamics: Predictable computational loads eliminate burst-induced latency spikes while ensuring step-time consistency in distributed training environments.

Complementing the C3PO framework, we utilize entropy regularization (He et al., 2025a) to the loss function, explicitly penalizing low-variance action distributions and thereby encouraging exploration of the policy model.

$$\mathbb{H}(\theta) = \mathcal{H}(\pi_\theta(\cdot \mid y_{i,<t})). \tag{4}$$

Figure 3 is the comparsion of our C3PO Strategy with the widely-used Dynamic Sampling Strategy (Yu et al., 2025). Each line represents a batch of grouped responses generated from the policy model for a single training step. In each group, there are the same number of responses, each composed of variant tokens. In each training step, all tokens are aggregated to form a token level global batch, which is then fed into optimizers such as AdamW (Loshchilov and Hutter, 2019). Due to the variable length of total tokens, the optimizer faces challenges as the gradients exhibit high variance, resulting in convergence difficulties. Previous approaches, such as dynamic sampling (Yu et al., 2025; Team et al., 2025), operate at the sample level by filtering and removing samples, yet fail to adequately address this class of problems. While C3PO operates at the token level by sampling tokens to form a token level global batch, each training step maintains consistent token input to optimizer. This results in reduced gradient variance and consequently achieves significantly more stable optimization.

Our model `Ring-lite` adopts MoE architecture, which fundamentally differs from conventional dense models due to its inclusion of multiple specialized experts. However, this design introduces challenges related to expert imbalance during training. To enhance training efficiency and to prevent imbalances in token distributions across experts, we incorporate load balance loss and router z-loss detailed in Ling-Team (2025), which is formulated as:

$$\mathbb{B}(\theta) = \frac{1}{N_e} \sum_{i=1}^{N_e} P_i * F_i * N_e, \quad \mathbb{Z}(\theta) = \frac{1}{M} \sum_{i=1}^{M} \Big(\log \sum_{j=1}^{N_e} \exp(z_{ij}) \Big)^2, \tag{5}$$

where N_e is the number of experts, M is the number of tokens, P_i and F_i are the average probability and count of the i-th expert selected across all tokens in a batch, and z_{ij} is the logits of the router.

In summary, the overall loss for training is formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{C3PO}}(\theta) + \alpha_{entropy} * \mathbb{H}(\theta) + \alpha_{balance} * \mathbb{B}(\theta) + \alpha_{zloss} * \mathbb{Z}(\theta) \tag{6}$$

3.3 Reward Model

3.3.1 Math & Science Verifier

Our training framework incorporates rule-based verifiable rewards in reinforcement learning, which has been proven effective for advancing the reasoning ability of large language models (DeepSeek-AI, 2025). For mathematical and scientific tasks, we append a brief instruction prompt after each input query to facilitate long chain-of-thought reasoning, *i.e.*, `Please reason step by step, and put your final answer within \\boxed{}`. We employ external verification tool, *i.e.*, Math-Verify [4], to evaluate the correctness of model responses. Specifically, a score of 1 is awarded for responses that correctly match the ground-truth answers, and a score of 0 is assigned to incorrect solutions. Since Math-Verify provides robust parsing ability that well accommodates various mathematical notations and expressions, we do not include any explicit format-related reward in our training framework.

3.3.2 Code Verifier

For code task, we build a code sandbox for reward verification, supporting code execuation and online judgeg tasks across several programming languages (e.g., Python, C++, Java). It offers multiple execution modes (function calls, online judging, unit testing) and interaction paradigms (real-time SDK/API for training, offline batch processing for data cleaning), achieving 8K/s throughput with sub-second latency. With the code sandbox, we employs a sparse outcome reward for RL training on code tasks. Specifically, the reward is defined based on the execuation results from the sandbox, *i.e.*,

$$R_{\text{code}} = \begin{cases} 1, & \text{All test cases passed} \\ 0, & \text{Otherwise} \end{cases}$$

It's worth noting that the reward mechanism employs a sparse design, wherein a reward of 1 is exclusively granted only if the code successfully passes all test cases; otherwise, the reward remains

[4]https://github.com/huggingface/Math-Verify

0. This approach stands in stark contrast to incremental reward systems that offer partial credit for solutions that are incomplete or only partially correct. By adopting this strategy, we ensure that models are incentivized to gain a thorough understanding of the problem, rather than focusing on superficial test cases. This prevents models from simply regurgitating answers to public test cases or overfitting to trivial edge cases, encouraging a more robust and well-rounded approach to problem-solving.

3.4 Training Pipeline

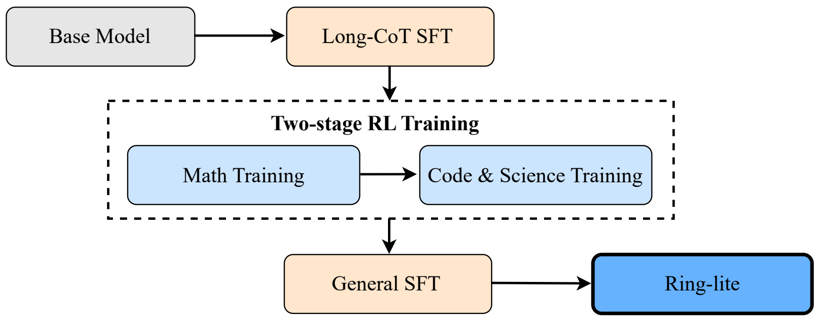


Figure 4 The training pipeline of Ring-lite

The overview of the training pipeline is depicted in Figure 4. It consists of a four-stage training process. We first conduct Long Chain-of-Thought Supervised Fine-Tuning (Long-CoT SFT) to obtain our distilled model, *i.e.*, `Ring-lite-distill`. This stage aims to directly distill the reasoning ability of a larger teacher model into our small-sized base model. From our preliminary analysis, we find that with our meticulously curated reasoning data, the reasoning ability of the distilled model can be further enhanced. However, directly applying reinforcement learning (RL) on mixed reasoning data is vulnerable to domain conflict, resulting in performance declines. Thus, we propose adopting a two-stage RL training pipeline: first, we run RL training on a math dataset, then incorporate code and science datasets in subsequent RL training. This approach empirically demonstrates that it can effectively preserve reasoning abilities across diverse fields. As both Long-Cot SFT and the two-stage RL training focus on improving performance on reasoning tasks, we additionally include a general SFT stage to enhance the model's ability in various general tasks, such as instruction following, creative writing, safety, and etc.

4 Experiment

4.1 Experimental Setup

4.1.1 Training Settings

As introduced in the training pipeline, to enhance the model's reasoning capabilities, we performed SFT on Ling-lite-1.5[5] (Ling-Team, 2025) using the well-constructed Long-CoT dataset. For optimization, we employed the AdamW optimizer with a weight decay of 0.1 and a learning rate of 3e-4, following a cosine decay schedule with a 1% linear warmup. The training configuration included a batch size of 256 over 3 epochs. To facilitate long-context reasoning, we set the context window of the model to 32,768 tokens and adjusted the RoPE base to 600,000 for improved stability.

[5]https://huggingface.co/inclusionAI/Ling-lite-1.5

In RL training with C3PO, we use a batch size L of 512, sampling $K = 16$ responses per prompt and adopt a learning rate of $3e-6$ with AdamW optimizer. The token-budget parameter is set to 409600. The maximum total length is configured to 24576 and is extended to 32768 in the second stage of code & science training. We set entropy loss coefficient $\alpha_{ent} = 5e-4$, load balance loss coefficient $\alpha_{balance} = 1e-5$, router z-loss coefficient $\alpha_{zloss} = 1e-7$, KL loss coefficient $\beta = 1e-3$. All experiments were performed on 256 * NVIDIA H800.

4.1.2 Benchmarks

For a comprehensive evaluation of the quality of our reasoning models, we implemented automatic benchmarks to assess their performance, which are categorized into the following dimensions.

- **Math:** MATH-500 (Lightman et al., 2024), AIME 2024, AIME 2025 (AIME, 2025), CNMO 2024[6], Livemathbench (Liu et al., 2024), MinervaMath (Lewkowycz et al., 2022).
- **Coding:** LiveCodeBench (Jain et al., 2025)(202408 – 202501), Codeforces[7].
- **Science:** GPQA Diamond (Rein et al., 2023), OlympiadBench (He et al., 2024).

4.1.3 Baselines

We conduct comprehensive evaluations against several baselines of similar parameter sizes, including Qwen3-8B-Thinking (Yang et al., 2025), R1-Distill-Qwen-7B, R1-Distill-Qwen-14B (DeepSeek-AI, 2025), AceReason-Nemotron-7B (Chen et al., 2025) and Ring-lite-distill-preview (Tang et al., 2025).

4.1.4 Evaluation Settings

For all reasoning models, we utilize a temperature of 1.0, a top-p value of 0.95, and a maximum output length of 32,768 tokens. In addition, the prompts are unified with the zero-shot setting. For mathematics benchmarks, we use Math-Verify as the evaluator to score model generations.

4.2 Main Results

The evaluation results are shown in Table 1. To provide a fair comparison, we evaluate our `Ring-lite` against recent competitive reasoning models with approximately 10B parameters. As shown in Table 1, our `Ring-lite` achieves the best average score across multiple reasoning tasks while using only 2.75B active parameters. This establishes our `Ring-lite` as the new state-of-the-art reasoning model among small-scale MoE models, offering performance comparable to or even surpassing that of the latest strong reasoning dense model under 10B parameters, *i.e.*, Qwen3-8b-Thinking. Additionally, compared to our previously released distilled MoE model, Ring-lite-distill-preview, our `Ring-lite` significantly improves reasoning performance on all benchmarks, further demonstrating the superiority of our training pipeline.

4.3 Key Findings

In this section, we analyze the central observations derived from training reinforcement learning across diverse domains, with particular focus on emergent instability phenomena, training efficiency between SFT and RL methodologies, and inter-domain data conflicts. To ensure a fair

[6] https://www.cms.org.cn/Home/comp/comp/cid/12.html

[7] The Codeforces was assessed through problems from 14 Div. 2 contests of Codeforces, combined with expert-designed test cases, followed by the computation of expected ratings and competitor proportions.

Table 1 The comparison of different reasoning models. The best and second-best results are in **bold** and underlined. Results with * are collected from their original papers. All models are evaluated with the same evaluation setting.

		R1-Distill -Qwen-7B	R1-Distill -Qwen-14B	AceReason -Nemotron-7B	Qwen3-8b -Thinking	Ring-lite -distill-preview	`Ring-lite`
	Architecture	Dense	Dense	Dense	Dense	MoE	MoE
	# Activated Params	7B	14B	7B	8B	2.75B	2.75B
	# Total Params	7B	14B	7B	8B	16.8B	16.8B
Math	MATH500$_{pass@1}$	92.80*	93.90*	94.10*	**97.40***	93.55	96.65
	AIME24$_{pass@1}$	55.50*	69.70*	69.00*	76.00*	57.81	**76.61**
	AIME25$_{pass@1}$	39.43	52.29	53.60*	67.30*	39.38	**69.11**
	CNMO24$_{pass@1}$	62.50	70.14	71.61	74.57	60.68	**75.61**
	LiveMathBench$_{pass@1}$	74.62	79.13	79.37	81.90	72.37	**83.74**
Coding	LiveCodeBench$_{pass@1}$	37.60*	53.10*	51.80*	58.06	32.62	**60.66**
	CodeForces$_{percentile}$	46.55	80.02	84.03	79.44	48.62	**86.45**
Science	GPQA Diamond$_{pass@1}$	49.10*	59.10*	50.67	**62.00***	52.81	61.05
	OlympiadBench$_{pass@1}$	70.26	74.86	75.95	79.65	68.74	**79.95**
	Average	58.71	70.25	70.01	75.15	58.51	**76.65**

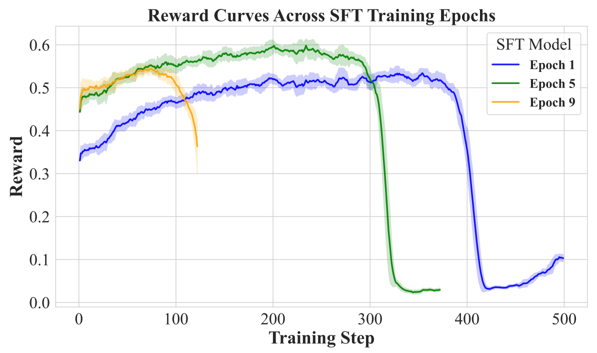
comparison, we applied supervised fine-tuning to the Qwen2.5-7B-Base (Qwen et al., 2025) model using our Long-CoT dataset as stated in Section 2.1, resulting in the `Ring-distill-Qwen-7B` model.

4.3.1 Move Towards Stable and Efficient RL

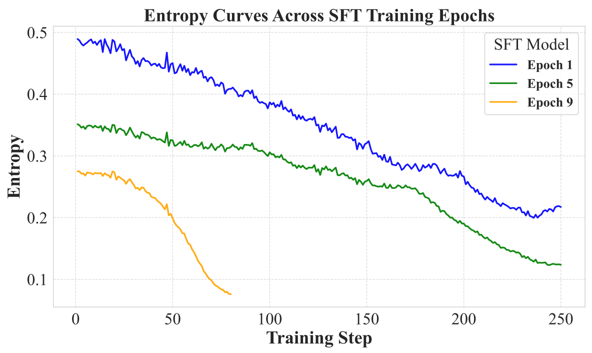
As outlined in the methodology, we empirically observe significant training instability and throughput fluctuations in GRPO during RL training. Here we present a systematic experimental evaluation confirming these phenomena, followed by a quantitative analysis establishing the effectiveness of our proposed method in addressing both challenges:

1. **Reward Collapse Phenomenon in RL Training on Distilled Models**

During reinforcement learning training with distilled models, we found that reward trajectories exhibit a precipitous decline after a few training steps, failing to recover to baseline levels and culminating in complete training collapse. Through rigorous empirical diagnostics, we identify two critical factors governing RL training stability: **Model Entropy**, which quantifies policy degradation in distilled models, and **Response Length Fluctuation**, a measure of sequence generation instability. These factors demonstrate strong correlation with reward collapse, as evidenced by quantitative ablation studies as follows:



(a) Ring-lite Reward



(b) Ring-lite Entropy Loss

Figure 5 The reward and entropy curves of Ring-lite.

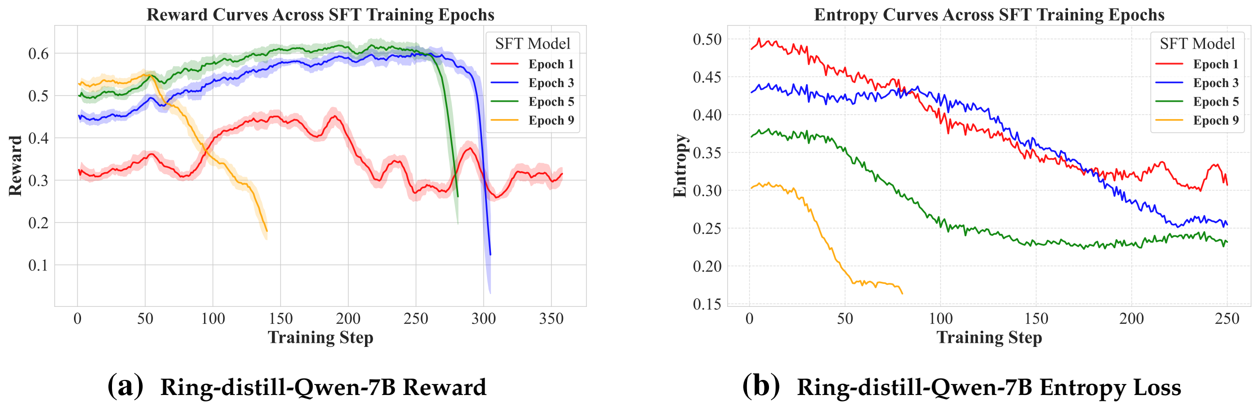


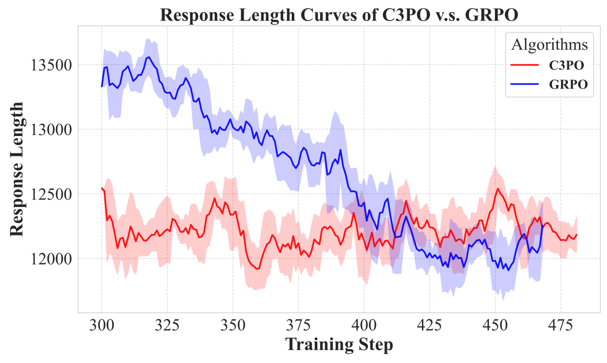
Figure 6 The reward and entropy curves of Ring-distill-Qwen-7B

- **Model Entropy:** As shown in Figure 5, the reward collapse during RL training exhibits a systematic dependence on the number of Long-CoT SFT epochs: models trained with more SFT epochs experience collapse earlier. Specifically, models trained with a greater number of SFT epochs exhibit earlier onset of reward collapse. This trend is accompanied by a concurrent reduction in entropy loss (Figure 6), revealing a robust inverse correlation between the magnitude of entropy loss and stability during RL training. Collectively, these results underscore that lower entropy loss during SFT corresponds to a higher propensity for reward collapse in subsequent RL phases, suggesting a statistically significant inverse relationship between these variables. Notably, this pattern persists across both MoE and dense model architectures, indicating architectural invariance in the observed phenomena.
- **Response Length Fluctuation:** Figure 7 (Ring-lite) and Figure 8 (Ring-distill-Qwen-7B) demonstrate that the generation length exhibits great variability across training steps, resulting in significant fluctuations in training token sizes. These unstable token training sizes greatly affect optimization stability, as evidenced by pronounced increases and occasional spikes—in gradient norms, leading to catastrophic reward collapse. This observation underscores the imperative need for developing strategies that mitigate both entropy loss and generation length variability to ensure stable RL training.

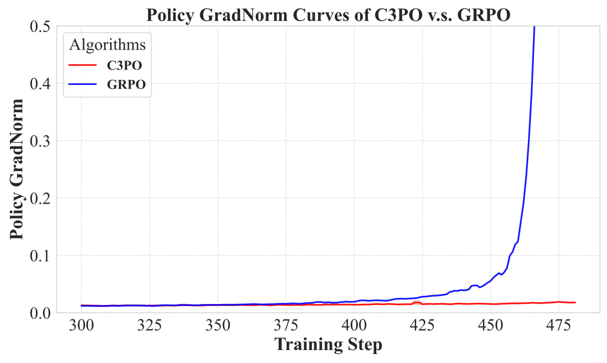
2. RL Training Throughput Fluctuation

Besides the challenge of reward collapse, our empirical observations reveal substantial throughput fluctuations emerging during RL training, which present considerable challenges for optimizing training efficiency in distributed systems. As shown in Figure 7 (Ring-lite) and Figure 8 (Ring-distill-Qwen-7B), these variations are primarily attributable to response length variability. Specifically, longer response sequences necessitate prolonged computation per training step, whereas shorter sequences underutilize computational resources, thereby diminishing throughput efficiency. This dynamic throughput behavior introduces significant optimization challenges in system design, as unpredictable computational demands complicate the implementation of efficient resource allocation and scheduling.

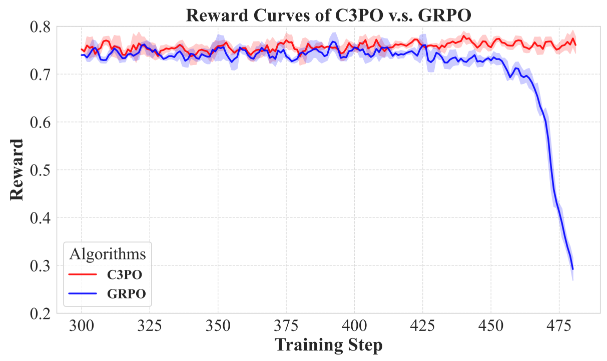
Since GRPO method suffers from the training instability and throughput fluctuation problems, C3PO is proposed to address these limitations through two key mechanisms: (1) selecting SFT checkpoints associated with higher entropy loss to stabilize optimization dynamics, and (2) imposing a fixed token budget during training to ensure training token consistency. To validate the



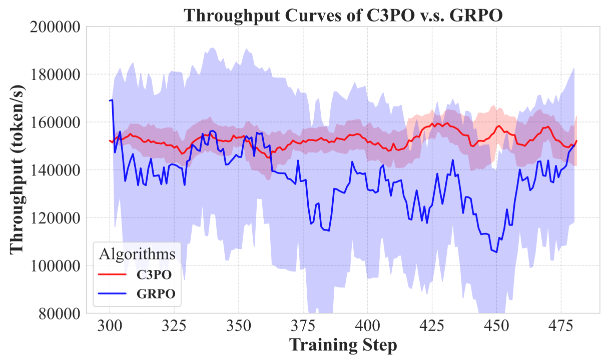
(a) Response-Length



(b) Gradient-Norm



(c) Rewards



(d) Throughput

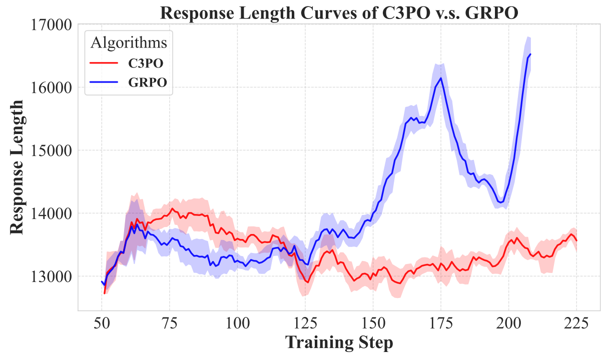
Figure 7 Training Dynamics of Ring-lite

efficacy of our method, we conduct experiments using two distilled initialization models with different model architectures: Ring-lite-distill and Ring-distill-Qwen-7B.

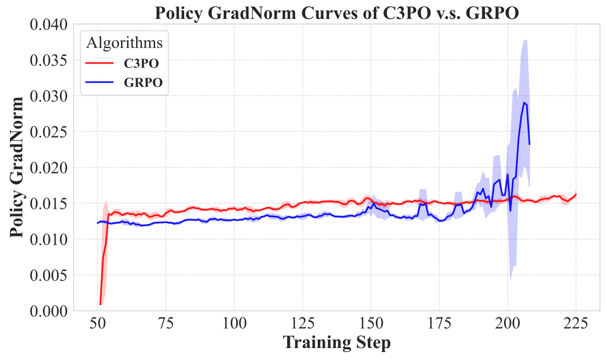
As illustrated in Figures 7 and 8, models trained with fewer epochs exhibit enhanced stability during RL training. However, our empirical analysis (Table 2) reveals a critical trade-off between training stability and final model performance. While models initialized with Epoch 1 checkpoints demonstrate superior stability in both configurations, their performance metrics lag significantly behind those of Epoch 7 and Epoch 3. Conversely, Epoch 9 achieves the highest initial performance but suffer from destabilization during later RL training phases, ultimately failing to surpass the results of Epoch 7 and Epoch 3. Furthermore, our methodological innovation in maintaining fixed training token size enables C3PO to consistently outperform GRPO across four critical metrics: generation length stability, gradient stability, reward stability, and throughput stability (Figures 7 and 8). In short, our C3PO not only resolves the instability inherent in GRPO on distilled models but also ensures efficient RL training, thereby bridging the gap between training robustness and model capability.

4.3.2 From Distill to RL: The Art of Balancing Token Efficiency

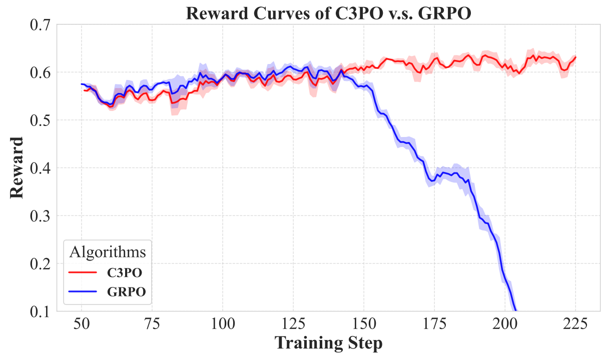
For our experiments, we used a constant warm-up learning rate scheduler with rates [1e-6, 3e-6], using the AdamW optimizer. Specifically, Ring-MoE performed better with a learning rate of 3e-6, while Qwen achieved better results at 2e-6. We utilized a training batch size of 512 prompts, a minibatch size of 2, and generated 16 responses for each prompt. In both the rollout and evaluation phases, the temperature was set to 1.0 to promote response diversity. For all methods, we set the



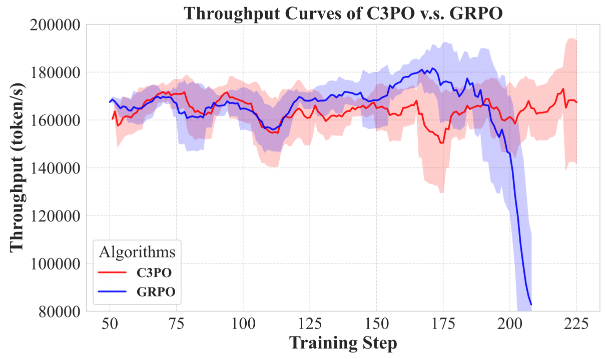
(a) Response-Length



(b) Gradient-Norm



(c) Rewards



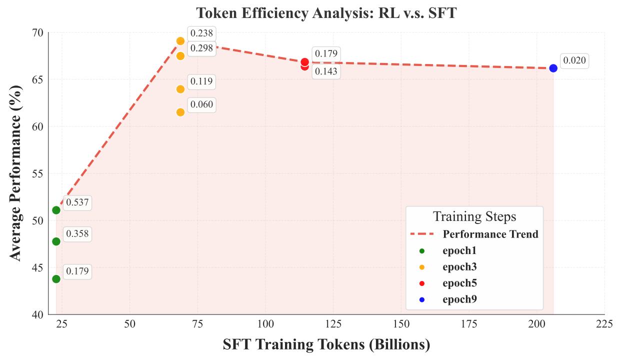
(d) Throughput

Figure 8 Training Dynamics of Ring-distill-Qwen-7B

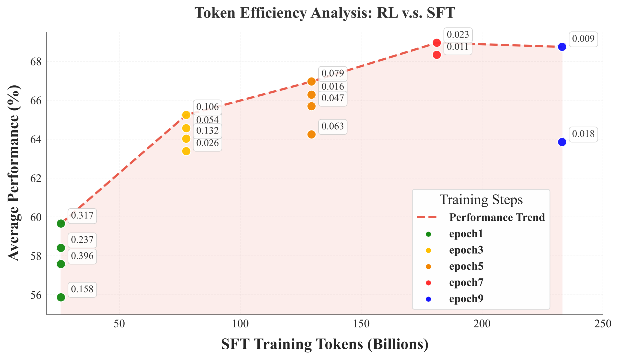
bound ϵ to 0.2, the KL coefficient was set to 0.001. The maximum total length is configured to 24576. All experiments were performed on 256 * NVIDIA H800.

To further validate the generalizability of our findings, we conducted additional experiments on the Qwen series of models (Bai et al., 2023), which have emerged as highly influential in the open-source community. Following the same methodology applied to `Ring-lite`, we first fine-tuned the Qwen2.5-7B-Instruct model using our Long-CoT dataset and subsequently performed RL training. We find that while distillation is effective, it requires significantly more training tokens than reinforcement learning (RL) to achieve comparable performance. Empirically, selecting checkpoints based on entropy loss within the range of 0.3-0.5 yields optimal results on our RL training setting. Entropy loss values below this threshold limit model exploration and reduce the chances of learning to solve more challenging problems, whereas excessive entropy loss leads to slower convergence and degraded model performance.

From Figure 5b, we observe that varying the number of training epochs of the distilled model significantly influences the trend of entropy loss, thereby determining the exploration scope for RL. Based on our experiments, increasing the number of SFT training epochs leads to a rapid collapse of entropy. However, insufficient SFT training inevitably results in inferior performance. To systematically quantify the choice of optimal SFT epoch, we employ token efficiency, *i.e.*, $\frac{\text{\# RL training tokens}}{\text{\# SFT training tokens}}$, to evaluate the relationships among RL training steps, SFT training steps, and average downstream performance. As shown in Figure 9 and Table 2, the best performance is achieved with a moderate number of SFT training epochs and suitable token efficiency. In our training pipeline, we utilize these findings to select the optimal SFT model.



(a) **The token efficiency on Ring-distill-Qwen-7B.**



(b) **The token efficiency analysis on Ring-lite.**

Figure 9 Reward curves across different SFT training epochs: (a) Ring-distill-Qwen-7B, (b) Ring-lite. Dots with the same color denote different values of token efficiency on the same SFT model.

Table 2 The comparison of RL training across different SFT epochs. The best results are in **bold**. $\triangle_{impr}$ denotes the average performance improvements compared to respective SFT models.

Model	SFT Epochs	AIME24@32	AIME25@32	GPQA	LiveCodeBench	Average	$\triangle_{impr}$
`Ring-distill-Qwen-7B`	Epoch 1	32.97	31.67	45.33	36.42	36.60	+10.19
	Epoch 3	62.45	49.01	57.17	48.61	**54.31**	**+11.03**
	Epoch 5	61.51	50.42	57.29	45.25	53.62	+5.62
	Epoch 9	56.35	44.64	56.22	46.86	51.02	+0.13
`Ring-lite`	Epoch 1	51.35	38.91	53.6	40.32	46.05	+3.03
	Epoch 5	63.54	42.08	53.88	44.13	50.91	**+8.76**
	Epoch 7	64.74	43.12	57.61	46.82	**53.07**	+7.27
	Epoch 9	65.73	43.85	56.91	45.70	53.05	+7.93

4.3.3 Resolving Domain Data Conflict: Beyond Mixed Solutions

For our experiments, we used a constant warm-up learning rate scheduler with rates [2e-6, 3e-6], using the AdamW optimizer. Specifically, Ring-MoE performed better with a learning rate of 3e-6, while Qwen achieved better results at 2e-6. We utilized a training batch size of 512 prompts, and generated 16 responses for each prompt. In both the rollout and evaluation phases, the temperature was set to 1.0 to promote response diversity. For all methods, the KL coefficient was set to 0.001. The maximum total length is configured to 24576. All experiments were performed on 256 * NVIDIA H800.

In our preliminary reinforcement learning (RL) experiments, we observed significant performance declines across various reasoning benchmarks when training our cold-start supervised fine-tuning (SFT) model with a combination of math and code reasoning datasets. We then conducted RL experiments on two representative distilled dense models: `DeepSeek-R1-Distill-Qwen-7B` and `Ring-distill-Qwen-7B`.

As shown in Table 3, combining reasoning datasets from the math and code domains does not lead to performance gains across different fields. Instead, the mixed dataset fails to outperform models trained exclusively on either math or code datasets. Notably, the experimental findings derived from our distilled models reveal that math-only training achieves superior performance on coding benchmarks compared to code-only, irrespective of the model's architectural configuration. However, this observation does not extend to the DeepSeek-derived models, indicating that the performance of RL training may be strongly influenced by the Long-CoT data in the SFT period. Conversely, code-only RL does not provide additional improvements for math tasks.

These results indicate that mixing diverse reasoning domains may introduce conflicts that hinder overall performance. Specialized training on individual domains appears to be more effective for optimizing performance in each specific area.

Table 3 The comparison of different training stages on Ring-lite, Qwen and DeepSeek distilled model. The best results are in **bold**, the performance differences compared to the best performance are denoted with arrows and numbers.

Model	Training Stages	AIME24@32	AIME25@32	GPQA	LiveCodeBench
`DeepSeek-R1-Distill-Qwen-7B`	Math-only	**59.64**	**44.22**	**49.94**	39.74
	Code-only	55.62	41.77	48.93	**42.56**
	Math & Code Mixed	58.44$\downarrow_{1.20}$	43.80$\downarrow_{0.42}$	49.81$\downarrow_{0.13}$	42.20$\downarrow_{0.36}$
`Ring-distill-Qwen-7B`	Math-only	**66.77**	**57.60**	58.18	**50.72**
	Code-only	55.21	45.68	55.30	47.94
	Math & Code Mixed	62.86$\downarrow_{3.91}$	54.22 $\downarrow_{3.38}$	**58.74**	49.73$\downarrow_{0.99}$
`Ring-lite`	Math-only	**78.54**	**67.60**	61.61	**61.11**
	Code-only	73.12	64.74	61.27	59.27
	Math & Code Mixed	76.77$\downarrow_{1.77}$	66.61$\downarrow_{0.99}$	**61.80**	60.44$\downarrow_{0.67}$

To enhance overall reasoning performance across diverse areas when training with multiple domain-specific datasets, we divided the our RL training into multiple stages. Specifically, we first conducted RL experiments using only the math dataset, followed by applying RL with scientific and code datasets. As shown in Table 4, our two-stage training strategy significantly improved downstream performance on challenging reasoning benchmarks, such as AIME25 and LiveCodeBench. Additionally, by doubling the amount of code and scientific training data, we achieved an average performance increase of 1% on both math and scientific benchmarks. Based on these results, we adopted this two-stage training strategy for `Ring-lite` to maintain superior overall reasoning abilities across multiple domains.

Table 4 The comparison of different training strategies.

Training Strategy	AIME24@32	AIME25@32	GPQA	LiveCodeBench
Naive Mixed RL	76.20	64.06	61.71	60.04
Two-stage RL	77.19 $\uparrow_{0.99}$	65.73$\uparrow_{1.67}$	61.48$\downarrow_{0.23}$	61.87$\uparrow_{1.83}$
Increase code & stem data				
Naive Mixed RL	75.94	65.31	60.54	59.81
Two-stage RL	78.54$\uparrow_{2.60}$	67.71$\uparrow_{2.40}$	61.93$\uparrow_{1.39}$	61.78$\uparrow_{1.97}$

5 Conclusion

This work introduces `Ring-lite`, a novel reasoning MoE model that achieves state-of-the-art performance across diverse challenging tasks. By integrating the proposed reinforcement learning method C3PO, `Ring-lite` significantly enhances reasoning capabilities while maintaining computational efficiency and stability. Empirical results demonstrate its efficacy, achieving scores of 76.61% (AIME24), 69.11% (AIME25), 60.66% (LiveCodeBench), 86.45% (Codeforces) and 61.05% (GPQA-diamond). Our research pursues two primary objectives: first, to refine RL training through algorithm-engineered co-design for enhanced efficiency; second, to address complex tasks requiring advanced reasoning to extend the boundaries of model intelligence.

In the future, we plan to investigate general reward modeling to attain human-verifier-level accuracy and explore systematic data synthesis techniques for scalable dataset expansion, thereby improving generalization capabilities.

6 Contributors

Authors are listed **alphabetically by the first name**.

Ling Team
Bin Hu
Cai Chen
Deng Zhao
Ding Liu
Dingnan Jin
Feng Zhu
Hao Dai
Hongzhi Luan
Jia Guo
Jiaming Liu
Jiewei Wu
Jun Mei
Jun Zhou
Junbo Zhao
Junwu Xiong
Kaihong Zhang
Kuan Xu
Lei Liang
Liang Jiang
Liangcheng Fu
Longfei Zheng
Qiang Gao
Qing Cui
Quan Wan
Shaomian Zheng
Shuaicheng Li
Tongkai Yang
Wang Ren
Xiaodong Yan
Xiaopei Wan
Xiaoyun Feng
Xin Zhao
Xinxing Yang
Xinyu Kong
Xuemin Yang
Yang Li
Yingting Wu
Yongkang Liu
Zhankai Xu
Zhenduo Zhang
Zhenglei Zhou
Zhenyu Huang
Zhiqiang Zhang
Zihao Wang
Zujie Wen

References

AIME. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIMEProblemsandSolutions, 2025.

Syeda Nahida Akter, Shrimai Prabhumoye, Matvei Novikov, Seungju Han, Ying Lin, Evelina Bakhturi, Eric Nyberg, Yejin Choi, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-crossthink: Scaling self-learning beyond math reasoning. *arXiv preprint arXiv: 2504.13941*, 2025.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. https://arxiv.org/abs/2502.17387.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. https://arxiv.org/abs/2309.16609.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning, 2025. https://arxiv.org/abs/2505.16400.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. https://arxiv.org/abs/2501.12948.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3828–3850. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.211. https://doi.org/10.18653/v1/2024.acl-long.211.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025a.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025b. https://arxiv.org/abs/2504.11456.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. *NeurIPS*, 2021.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025. https://arxiv.org/abs/2503.24290.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. https://openreview.net/forum?id=chfJJYC3iL.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset, 2023. https://arxiv.org/abs/2312.14852.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode, 2022. https://www.science.org/doi/abs/10.1126/science.abq1158.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. https://openreview.net/forum?id=v8L0pN6EOi.

Ling-Team. Every flop counts: Scaling a 300b mixture-of-experts ling llm without premium gpus, 2025. https://arxiv.org/abs/2503.05139.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. Are your llms capable of stable reasoning? *CoRR*, abs/2412.13147, 2024. doi: 10.48550/ARXIV.2412.13147. https://doi.org/10.48550/arXiv.2412.13147.

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.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. https://arxiv.org/abs/1711.05101.

Dakuan Lu, Xiaoyu Tan, Rui Xu, Tianchu Yao, Chao Qu, Wei Chu, Yinghui Xu, and Yuan Qi. Scp-116k: A high-quality problem-solution dataset and a generalized pipeline for automated extraction in the higher education science domain, 2025. https://arxiv.org/abs/2501.15587.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025a. https://www.together.ai/blog/deepcoder.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025b. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2.

OpenAI. Openai o1 system card, 2024. https://arxiv.org/abs/2412.16720.

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, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. https://arxiv.org/abs/2412.15115.

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

Caizhi Tang, Chilin Fu, Chunwei Wu, Jia Guo, Jianwen Wang, Jingyu Hu, Liang Jiang, Meng Li, Peng Jiao, Pingping Liu, Shaomian Zheng, Shiwei Liang, Shuaicheng Li, Yalin Zhang, Yingting Wu, Yongkang Liu, and Zhenyu Huang. Holistic capability preservation: Towards compact yet comprehensive reasoning models, 2025. https://github.com/inclusionAI/Ring/blob/main/Ring_Lite_Distill_Preview.pdf.

Tencent Hunyuan Team, Ao Liu, Botong Zhou, Can Xu, Chayse Zhou, ChenChen Zhang, Chengcheng Xu, Chenhao Wang, Decheng Wu, Dengpeng Wu, Dian Jiao, Dong Du, Dong Wang, Feng Zhang, Fengzong Lian, Guanghui Xu, Guanwei Zhang, Hai Wang, Haipeng Luo, Han Hu, Huilin Xu, Jiajia Wu, Jianchen Zhu, Jianfeng Yan, Jiaqi Zhu, Jihong Zhang, Jinbao Xue, Jun Xia, Junqiang Zheng, Kai Liu, Kai Zhang, Kai Zheng, Kejiao Li, Keyao Wang, Lan Jiang, Lixin Liu, Lulu Wu, Mengyuan Huang, Peijie Yu, Peiqi Wang, Qian Wang, Qianbiao Xiang, Qibin Liu, Qingfeng Sun, Richard Guo, Ruobing Xie, Saiyong Yang, Shaohua Chen, Shihui Hu, Shuai Li, Shuaipeng Li, Shuang Chen, Suncong Zheng, Tao Yang, Tian Zhang, Tinghao Yu, Weidong Han, Weijie Liu, Weijin Zhou, Weikang Wang, Wesleye Chen, Xiao Feng, Xiaoqin Ren, Xingwu Sun, Xiong Kuang, Xuemeng Huang, Xun Cao, Yanfeng Chen, Yang Du, Yang Zhen, Yangyu Tao, Yaping Deng, Yi Shen, Yigeng Hong, Yiqi Chen, Yiqing Huang, Yuchi Deng, Yue Mao, Yulong Wang, Yuyuan Zeng, Zenan Xu, Zhanhui Kang, Zhe Zhao, ZhenXiang Yan, Zheng Fang, Zhichao Hu, Zhongzhi Chen, Zhuoyu Li, Zongwei Li, Alex Yan, Ande Liang, Baitong Liu, Beiping Pan, Bin Xing, Binghong Wu, Bingxin Qu, Bolin Ni, Boyu Wu, Chen Li, Cheng Jiang, Cheng Zhang, Chengjun Liu, Chengxu Yang, Chengzhong Xu, Chiyu Wang, Chong Zha, Daisy Yi, Di Wang, Fanyang Lu, Fei Chen, Feifei Liu, Feng Zheng, Guanghua Yu, Guiyang Li, Guohua Wang, Haisheng Lin, Han Liu, Han Wang, Hao Fei, Hao Lu, Haoqing Jiang, Haoran Sun, Haotian Zhu, Huangjin Dai, Huankui Chen, Huawen Feng, Huihui Cai, Huxin Peng, Jackson Lv, Jiacheng Shi, Jiahao Bu, Jianbo Li, Jianglu Hu, Jiangtao Guan, Jianing Xu, Jianwei Cai, Jiarong Zhang, Jiawei Song, Jie Jiang, Jie Liu, Jieneng Yang, Jihong Zhang, Jin lv, Jing Zhao, Jinjian Li, Jinxing Liu, Jun Zhao, Juntao Guo, Kai Wang, Kan Wu, Lei Fu, Lei He, Lei Wang, Li Liu, Liang Dong, Liya Zhan, Long Cheng, Long Xu, Mao Zheng, Meng Liu, Mengkang Hu, Nanli Chen, Peirui Chen, Peng He, Pengju Pan, Pengzhi Wei, Qi Yang, Qi Yi, Roberts Wang, Rongpeng Chen, Rui Sun, Rui Yang, Ruibin Chen, Ruixu Zhou, Shaofeng Zhang, Sheng Zhang, Shihao Xu, Shuaishuai Chang, Shulin Liu, SiQi Wang, Songjia Feng, Songling Yuan, Tao Zhang, Tianjiao Lang, Tongkai Li, Wei Deng, Wei Li, Weichao Wang, Weigang Zhang, Weixuan Sun, Wen Ouyang, Wenxiang Jiao, Wenzhi Sun, Wenzhuo Jia, Xiang Zhang, Xiangyu He, Xianshun Ren, XiaoYing Zhu, Xiaolong Guo, Xiaoxue Li, Xiaoyu Ma, Xican Lu, Xinhua Feng, Xinting Huang, Xinyu Guan, Xirui Li, Xu Zhang, Xudong Gao, Xun Luo, Xuxiang Qi, Yangkun Chen, Yangyu Tao, Yanling Xiao, Yantao Mai, Yanze Chen, Yao Ding, Yeting Yang, YiFan Song, Yifan Yang, Yijiao Zhu, Yinhe Wu, Yixian Liu, Yong Yang, Yuanjun Cai, Yuanlin Tu, Yue Zhang, Yufei Huang, Yuhang Zhou, Yuhao Jiang, Yuhong Liu, Yuhui Hu, Yujin Lin, Yun Yang, Yunhao Wang, Yusong Zhang, Zekun Wu, Zelong Zhang, Zhan Yu, Zhaoliang Yang, Zhe Zhao, Zheng Li, Zhenyu Huang, Zhiguang Liu, Zhijiang Xu, Zhiqing Kui, Zhiyin Zeng, Zhiyuan Xiong, Zhuo Han, Zifan Wu, Zigang Geng, Zilong Zhao, Ziyan Tang, Ziyuan Zhu, Zonglei Zhu, and Zhijiang Xu. Hunyuan-turbos: Advancing large language models through mamba-transformer synergy and adaptive chain-of-thought, 2025. https://arxiv.org/abs/2505.15431.

Xiong Jun Wu, Zhenduo Zhang, ZuJie Wen, Zhiqiang Zhang, Wang Ren, Lei Shi, Cai Chen, Deng Zhao, Qing Wang, Xudong Han, Chengfu Tang, Dingnan Jin, Qing Cui, and Jun Zhou. Sharp: Synthesizing high-quality aligned reasoning problems for large reasoning models reinforcement learning, 2025. https://arxiv.org/abs/2505.14147.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. https://arxiv.org/abs/2503.14476.

Xiaojiang Zhang, Jinghui Wang, Zifei Cheng, Wenhao Zhuang, Zheng Lin, Minglei Zhang, Shaojie Wang, Yinghan Cui, Chao Wang, Junyi Peng, Shimiao Jiang, Shiqi Kuang, Shouyu Yin, Chaohang Wen, Haotian Zhang, Bin Chen, and Bing Yu. Srpo: A cross-domain implementation of large-scale reinforcement learning on llm, 2025. https://arxiv.org/abs/2504.14286.