Walk Before You Run! Concise LLM Reasoning via Reinforcement Learning

Anonymous ACL submission

Abstract

As test-time scaling becomes a pivotal research frontier in Large Language Models (LLMs) development, contemporary and advanced post-training methodologies increasingly focus on extending the generation length of long Chain-of-Thought (CoT) responses to enhance reasoning capabilities toward DeepSeek R1-like performance. However, recent studies reveal a persistent overthinking phenomenon in state-of-the-art reasoning models, manifesting as excessive redundancy or repetitive thinking patterns. To address this issue, in this paper, we propose a simple yet effective two-stage reinforcement learning framework for achieving concise reasoning in LLMs, named **ConciseR**. Specifically, the first stage, using more training steps, aims to incentivize the model's reasoning capabilities via **G**roup **R**elative **P**olicy **O**ptimization with *clip-higher* and *dynamic sampling* components (**GRPO++**), and the second stage, using fewer training steps, explicitly enforces conciseness and improves efficiency via **L**ength-aware **G**roup **R**elative **P**olicy **O**ptimization (**L-GRPO**). ***Significantly, ConciseR only optimizes response length once all rollouts of a sample are correct, following the "walk before you run" principle.*** Experimental results show that our **ConciseR** model, which generates more concise CoT responses, outperforms recent state-of-the-art reasoning models with zero RL paradigm across AIME 2024, MATH-500, AMC 2023, Minerva, and Olympiad benchmarks. The code, training dataset, and model checkpoints will be publicly released.

1 Introduction

Test-time scaling (Snell et al., 2024; Muennighoff et al., 2025) has demonstrated a robust correlation between extending the generation length of Chain-of-Thought (CoT) (Wei et al., 2022) and improving the reasoning capabilities of Large Language Models (LLMs). The advent of large reasoning models, such as GPT-o1 (Jaech et al., 2024) and DeepSeek-

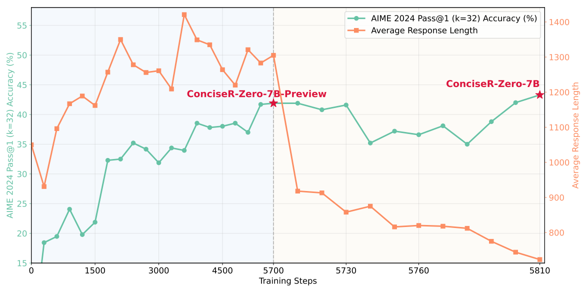

Figure 1: A detailed evaluation of accuracy and response length throughout the training steps.

R1 (Guo et al., 2025), represents a significant breakthrough in natural language processing, especially in tackling complex and intricate reasoning tasks. An interesting phenomenon observed during reinforcement learning post-training via Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is the emergence of an "aha moment" (Guo et al., 2025), which refers to a pivotal inflection point at which the model spontaneously initiates self-correction behaviors. These emergent behaviors develop autonomously through the model's exploration of the solution space rather than through explicit programming. Prior research (Yeo et al., 2025; Luo et al., 2025) has found a distinctive pattern following this moment: the response length of the model tends to increase significantly, accompanied by improvements in overall performance. Despite the lack of a clear understanding of why this occurs, this phenomenon has led many researchers to advocate for longer responses, leveraging additional computational resources in the hope of further enhancing accuracy.

However, generating excessively long CoT reasoning responses substantially increases computational overhead in both the model training and deployment phases. Furthermore, recent studies (Team et al., 2025; Cuadron et al., 2025) have discovered an intrinsic overthinking phenomenon in reasoning models, where these models persistently produce verbose rationalizations. This tendency manifests as the inclusion of irrelevant contextual information and unnecessary reflective behaviors. Such information and behaviors not only inefficiently consume computational resources but also compromise reasoning accuracy by causing models to deviate from valid logical pathways to incorrect conclusions.

To address these issues, recent studies (Luo et al., 2025; Song et al., 2025; Yu et al., 2025; Liu et al., 2025; Fatemi et al., 2025) are researching efficient reasoning methodologies based on the GRPO algorithm for training the model to produce more concise CoT responses, and have discovered a trade-off between the CoT response length and model reasoning capabilities in most cases, i.e., the shorter the length, the worse the performance. It is understandable that achieving efficient reasoning, improving ability via a more concise CoT, is inherently more challenging. This is in contrast to boosting performance by merely increasing the response length, as the former requires significantly higher model capabilities. Therefore, we highlight the critical importance of the timing for optimizing response length when training using GRPO-based algorithms. Adhering to the "walk before you run" principle, we consider that during training, response length optimization is only enabled when all rollouts for a training sample are correct.

Motivated by this, we propose ConciseR, which is a simple yet effective two-stage reinforcement learning framework for concise reasoning. Specifically, the first stage aims to enhance the model's reasoning capabilities through group relative policy optimization with clip-higher and dynamic sampling techniques. Meanwhile, we introduce the entropy bonus in the first stage, which is used to encourage exploration in policy gradient to incentivize the reasoning capabilities of the model further. The second stage enforces conciseness and improves efficiency via length-aware group relative policy optimization (incorporating max response length to calculate the length reward). Experiments demonstrate that ConciseR is compatible with RL models that incentivize reasoning, achieving reductions in response length while improving accuracy across benchmarks such as AIME 2024, AMC 2023, MATH-500, Minerva, and Olympiad datasets. As shown in Figure 1, ConciseR gradually activates the reasoning ability of the model in the first stage, then rapidly compresses the CoT response length in the second stage to achieve concise reasoning. Notably, the reduction in response length has immediate implications for computational efficiency, resource utilization, and response time, making the approach practically appealing and cost-effective.

2 Preliminary

2.1 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is one of the policy gradient methods that introduces a clipped surrogate objective for policy optimization. By using clipping to constrain policy updates within a proximal region of the previous policy, PPO stabilizes training and improves sample efficiency. Specifically, PPO updates the policy by maximizing the following objective:

$$\begin{aligned}\mathcal{J}_{\text{PPO}}(\theta) =&\mathbb{E}_{q\sim\mathcal{D},o_{\le t}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\\ &\min[\frac{\pi_\theta(o_t\mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t\mid q,o_{<t})}\hat{A}_t,\\ &\text{clip}(\frac{\pi_\theta(o_t\mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t\mid q,o_{<t})},1-\varepsilon,1+\varepsilon)\hat{A}_t],\end{aligned} \quad (1)$$

where $\pi_{\theta_{\text{old}}}$ is the policy before the update, ε is the clipping hyper-parameter, q indicates the question from the data distribution $\mathcal{D}$, and $\hat{A}_t$ is an estimator of the advantage function of the t-th token. Here, a standard and traditional way to estimate $\hat{A}_t$ is to compute the Generalized Advantage Estimation (GAE) (Schulman et al., 2017) with a learned value model. However, in the context of LLM RL scaling, learning the value model is computationally expensive, so methods that estimate $\hat{A}_t$ without a learned value model are practically preferred.

2.2 Group Relative Policy Optimization (GRPO)

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) introduces a policy gradient framework that eliminates the reliance on explicit value function by utilizing comparative advantage estimation within a group of responses. This method samples multiple candidate outputs for each input question and computing advantages based on the relative rewards among these candidates within their respective groups. Specifically, GRPO first samples a group of responses $\{o_1, o_2, \ldots, o_G\}$ per question and computes their returns $\mathbf{r} = \{r_1, r_2, \ldots, r_G\}$, then calculates the advantage of the i-th response o_i as,

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}. \quad (2)$$

Similar to PPO, GRPO uses a clipped objective with KL penalty and optimizes the policy model π_θ by maximizing the following objective:

$$
\begin{aligned}
\mathcal{J}_{\text{GRPO}}(\theta) =& \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{\text{G}}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \\
& \frac{1}{\text{G}}\sum_{i=1}^{\text{G}}\{\min[\tau_i(\theta)\hat{A}_i, \text{clip}(\tau_i(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_i] \\
& - \beta\mathbb{D}_{\text{KL}}[\pi_\theta||\pi_{\text{ref}}]\},
\end{aligned} \tag{3}
$$

where

$$
\begin{aligned}
\tau_i &= \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, \\
\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}}) &= \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - 1.
\end{aligned} \tag{4}
$$

Here, π_{ref} represents the reference model and the term $\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})$ indicates a KL penalty term to limit how much the trained model π_θ can deviate from the reference model π_{ref}.

2.3 Decouple Clip and Dynamic Sampling Policy Optimization (DAPO)

An in-depth analysis (Yu et al., 2025) reveals that the naive GRPO baseline suffers from several significant issues, such as entropy collapse, reward noise, and training instability. To address this issue, DAPO introduces four key techniques to make RL shine in the long-CoT RL scenario, including

- *Clip-Higher*, which enhances the diversity of the model and avoids entropy collapse.
- *Dynamic Sampling*, which improves training efficiency and stability.
- *Token-Level Policy Gradient Loss*, which plays a crucial role in reinforcement learning scenarios involving Long-CoT reasoning responses.
- *Overlong Reward Shaping*, a length-aware penalty mechanism designed to shape the reward for truncated samples to reduce reward noise and stabilize training.

Similar to GRPO, DAPO estimates the advantage in a group-relative manner and optimizes the policy model via the following objective,

$$
\begin{aligned}
\mathcal{J}_{\text{DAPO}}(\theta) = & \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{\text{G}}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\frac{1}{\text{G}}\sum_{i=1}^{\text{G}} \\
& \{\min[\tau_i(\theta)\hat{A}_i, \text{clip}(\tau_i(\theta), 1-\varepsilon_l, 1+\varepsilon_h)\hat{A}_i]\} \\
& \text{s.t.} \quad 0 < |\{o_i|\text{is\_equivalent}(o_i, a)\}| < \text{G},
\end{aligned} \tag{5}
$$

where ε_l and ε_h indicate the lower and higher clipping range.

3 Methodology

Our primary goal is to let LLM generate a more concise CoT response without sacrificing the model's performance. To this end, we propose a novel two-stage reinforcement learning training paradigm guided by the principle of "*Aim for 100% accuracy first; speed comes with mastery.*" Specifically, the first stage aims to incentivize the reasoning capabilities of the base model via group relative policy optimization with clip-higher and dynamic sampling, thus ensuring accuracy and robust model reasoning. Subsequently, the second stage explicitly enforces conciseness and improved efficiency via length-aware group relative policy optimization, aligning with our objective of achieving mastery through precision, where conciseness naturally follows from reliable and accurate reasoning.

3.1 Group Relative Policy Optimization with Clip-Higher and Dynamic Sampling (GRPO++)

In the first stage, the model is trained to incentivize the reasoning capabilities, which aims to enhance the model's problem-solving capacity, with an expected increase in response length as GRPO mostly encounters negative rewards and encourages the trained model toward longer responses. To this end, in this paper, we adopt GRPO with two key components of DAPO, clip higher and dynamic sampling, and further introduce an entropy bonus to encourage greater exploration capability in the model, named GRPO++. Similar to the original approach, GRPO++ estimates the advantage in a group-relative manner and optimizes the policy model using the following objective:

$$
\begin{aligned}
\mathcal{J}_{\text{GRPO++}}(\theta) = & \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{\text{G}}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \\
& \frac{1}{\text{G}}\sum_{i=1}^{\text{G}}\{\min[\tau_i(\theta)\hat{A}_i, \text{clip}(\tau_i(\theta), 1-\varepsilon_l, 1+\varepsilon_h)\hat{A}_i] \\
& + \alpha\mathbb{H}(\pi_\theta)\}, \\
& \text{s.t.} \quad 0 < |\{o_i|\text{is\_equivalent}(o_i, a)\}| < \text{G},
\end{aligned} \tag{6}
$$

where $\alpha\mathbb{H}(\pi_\theta)$ denotes the entropy bonus.

3.2 Length-Aware Group Relative Policy Optimization (L-GRPO)

Recent studies (Luo et al., 2025; Song et al., 2025) have found that the reasoning response length is not strongly correlated with the correctness of the answer; that is, a long CoT reasoning response does not necessarily represent a correct result, and a short CoT reasoning response does not necessarily

Table 1: Training Template. **{question}** will be replaced with the specific question during training.

A conversation between User and Assistant. The user asks a question, and the Assistant solves it.
The assistant first thinks about the reasoning process in the mind and then provides the user with the answer.
The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively,
i.e., <think> reasoning process here </think> <answer> answer here </answer>.
Please reason step by step, and put your final answer within \boxed{}.

User:
{question}

Assistant:

represent an incorrect one. On the contrary, the correct CoT reasoning responses are usually shorter in length, while incorrect reasoning responses tend to be longer.

Based on the above analysis, we reshape the reward function in GRPO. When the model's rollout results for a question are all correct, we further optimize the model's reasoning length for that question by using the remaining maximum response length as a reward (under the specified context length, the more remaining context length, the higher the reward), as calculated below.

$$\tilde{A}_i = \frac{\hat{r}_i - \mathrm{mean}(\{\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_{\mathrm{G}}\})}{\mathrm{std}(\{\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_{\mathrm{G}}\})}, \quad \hat{r}_i = r_i + \lambda \hat{\mathcal{L}}_i, \tag{7}$$

$$\hat{\mathcal{L}}_i = \begin{cases} 1 - \frac{\mathcal{L}_i}{\mathcal{L}_{\mathrm{Max}}}, & \text{if } \sum_{i=1}^{\mathrm{G}} r_i = \mathrm{G} \\ 0, & \text{if } \sum_{i=1}^{\mathrm{G}} r_i \neq \mathrm{G}. \end{cases} \tag{8}$$

Here, $\mathcal{L}_i$ is the length of the i-th response and $\mathcal{L}_{\mathrm{Max}}$ indicates the max response length. Then, we optimize the policy using the following objective:

$$\begin{aligned} \mathcal{J}_{\text{L-GRPO}}(\theta) &= \mathbb{E}_{q\sim\mathcal{D},\{o_i\}_{i=1}^{\mathrm{G}}\sim\pi_{\theta_{\mathrm{old}}}(\cdot|q)} \\ &\frac{1}{\mathrm{G}}\sum_{i=1}^{\mathrm{G}}\{\min[\tau_i(\theta)\hat{A}_i, \mathrm{clip}(\tau_i(\theta), 1-\varepsilon_l, 1+\varepsilon_h)\hat{A}_i] \\ &- \beta\mathbb{D}_{\mathrm{KL}}[\pi_\theta||\pi_{\mathrm{ref}}] + \alpha\mathbb{H}(\pi_\theta)\}, \\ &\text{s.t.} \quad 0 < |\{o_i|\mathrm{is\_equivalent}(o_i, a)\}|. \end{aligned} \tag{9}$$

3.3 Rule-based Reward Model

Using a trained reward model typically introduces the issue of reward hacking [24–29]. To mitigate this issue, we directly adopt the final accuracy from a verifiable task as the outcome reward, calculated according to the following rule:

$$r_i(o_i, a) = \begin{cases} 1, & \text{if is\_equivalent}(o_i, a) \\ 0, & \text{if not is\_equivalent}(o_i, a) \end{cases} \tag{10}$$

where a indicates the ground-truth answer and o_i contains the predicted answer. Additionally, it is important to note that the trained model must adhere strictly to the training prompt by generating the chain-of-thought within the <think></think> tags and subsequently presenting the answer within the <answer></answer> tags with the boxed tag.

3.4 Training Dataset Curation

To select and curate high-quality data for scaling RL, we include challenging problems from DeepScaleR (Luo et al., 2025), DAPO-Math-17K (Yu et al., 2025), and MATH (Hendrycks et al., 2021) to enhance problem difficulty and diversity in our data mixture:

DeepScaleR[1], which contains approximately 40K unique mathematics-specific problem-answer pairs collected from AIME (1984-2023), AMC (prior to 2023), Omni-MATH, and Still datasets (Lewkowycz et al., 2022; Gao et al., 2024; Min et al., 2024).

DAPO-Math-17K[2], which contains approximately 17K problem-answer pairs, each paired with an integer as the answer. DAPO-Math-17K was compiled from the Art of Problem Solving (AoPS[3]) website and official competition websites using a combination of web scraping and manual annotation.

MATH[4] (Level 3-5), which contains approximately 8K problem-answer pairs. Each problem has a step-by-step solution which can be used to teach models to generate explanations.

After obtaining the above datasets, we employ Math-Verify[5] to re-extract answers from the provided textual solutions, selecting only those cases where the extracted answer matches the corresponding answer in the dataset. We discard any samples that are empty, incomplete, or duplicates. Finally,

[1] https://huggingface.co/datasets/agentica-org/DeepScaleR-Preview-Dataset

[2] https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k

[3] https://artofproblemsolving.com/

[4] https://huggingface.co/datasets/EleutherAI/hendrycks_math

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

Table 2: Overall performance on five competition-level reasoning benchmarks. Our models outperform prior state-of-the-art models (7B) with zero RL paradigm. † indicate the results from (Hochlehnert et al., 2025).

Model	AIME 2024	MATH-500	AMC 2023	Minerva Math	Olympiad Bench	Avg. Score
Qwen2.5-7B-Base (Yang et al., 2024a)†	3.3	64.6	30.0	25.7	29.0	30.5
Qwen2.5-Math-7B-Base (Yang et al., 2024b)†	20.7	64.3	56.2	17.3	29.0	37.5
Qwen2.5-7B-Instruct (Yang et al., 2024a)†	12.3	77.1	52.8	34.9	38.7	43.2
Qwen2.5-Math-7B-Instruct (Yang et al., 2024b)†	15.7	82.9	67.0	35.0	41.3	48.4
Eurus-2-7B-PRIME (Cui et al., 2025)†	17.8	80.1	63.0	37.5	43.9	48.5
Open-Reasoner-Zero-7B (Hu et al., 2025)†	19.7	83.9	59.5	31.6	47.6	48.5
SimpleRL-Zero-7B (Zeng et al., 2025)†	14.0	77.9	58.0	33.0	39.0	44.4
SimpleRL-Zero-Math-7B (Zeng et al., 2025)†	22.7	76.9	62.2	30.1	39.3	46.2
Oat-Zero-7B (Liu et al., 2025)†	28.0	79.4	66.2	34.4	43.8	50.4
ConciseR-Zero-7B-Preview	42.8	83.0	73.9	31.8	45.1	55.3
ConciseR-Zero-7B	43.3	83.0	76.7	31.5	46.0	56.1

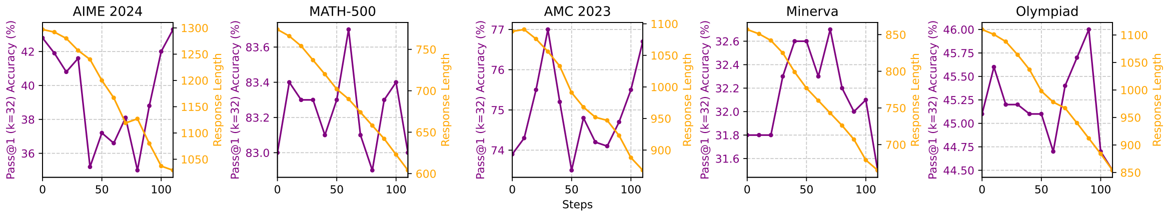

Figure 2: A detailed evaluation of accuracy and response length throughout the training steps.

we obtain approximately 59K reasoning problems as the training dataset. It should be noted that in the first stage, we use the 59K data to incentivize the model's reasoning ability. Still, in the second stage, we use the MATH (Level 3-5) data as the training set to optimize the model's reasoning length.

4 Experiments

4.1 Training Details

In this paper, we train our models using the verl[6] framework (Sheng et al., 2025) and leverage Qwen2.5-Math-7B (Yang et al., 2024b) as the base model. During training, we utilize the Adam [39] optimizer with a constant learning rate of 1×10^{-6}. We leverage a batch size of 128 with each question generating 32 rollouts, the maximum response length is set to 3,072 tokens, and training is conducted using mini-batches of size 128. As for the Clip-Higher, similar to the prior work (Yu et al., 2025), we set the clipping parameter ε_l to 0.2 and ε_h to 0.28, which effectively balance the trade-off between exploration and exploitation for RL. Specifically, for GRPO++, we set the entropy coefficient α to 0.001. For L-GRPO, we set the KL penalty coefficient β to 0.01 and set the λ to 0.000002.

[6] https://github.com/volcengine/verl

4.2 Evaluation Benchmarks

Similar to the prior work (Liu et al., 2025; Song et al., 2025), the performance of our models is evaluated on a diverse suite of competition-level benchmarks including AIME 2024[7] (comprises 30 challenge problems), AMC 2023[8] (contains 40 mathematical problems, covering algebra, geometry, number theory, and combinatorics), Minerva Math (Lewkowycz et al., 2022), MATH-500 (Hendrycks et al., 2021) (is a challenging benchmark comprising competition-level problems), and OlympaidBench (He et al., 2024).

4.3 Evaluation Setup

In this paper, our two-stage RL training framework aims to enhance the reasoning performance while reducing the response length, thereby enabling more concise reasoning. To this end, we adopt the Pass@k evaluation metric, reporting Pass@1 accuracy computed with a non-zero sampling temperature. Therefore, we set the maximum response length to 3,072 tokens. Specifically, we select a temperature of 0.6 combined with a top-p value of 0.95 to generate multiple responses (typically 32

[7] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

[8] https://huggingface.co/datasets/AI-MO/aimo-validation-amc

```
Question
There exist real numbers $x$ and $y$, both greater than 1, such that
$\log_x\left(y^x\right)=\log_y\left(x^{4y}\right)=10$. Find $xy$.

Response (a) (1023 characters)
To solve the problem, we start by analyzing the given equations:

1. \(\log_x(y^x) = 10\)
2. \(\log_y(x^{4y}) = 10\)

We can use the properties of logarithms to simplify these equations.
Recall that \(\log_a(b^c) = c \log_a(b)\). Applying this property, we get:

1. \(\log_x(y^x) = x \log_x(y) = 10\)
2. \(\log_y(x^{4y}) = 4y \log_y(x) = 10\)

From the first equation, we have:
\[ x \log_x(y) = 10 \implies \log_x(y) = \frac{10}{x} \]

From the second equation, we have:
\[ 4y \log_y(x) = 10 \implies \log_y(x) = \frac{10}{4y} = \frac{5}{2y} \]

Next, we use the change of base formula for logarithms, which states
that \(\log_a(b) = \frac{1}{\log_b(a)}\). So, \(\log_x(y) =
\frac{1}{\log_y(x)}\). Substituting the expressions we found, we get:
\[ \frac{10}{x} = \frac{1}{\frac{5}{2y}} = \frac{2y}{5} \]

This simplifies to:
\[ \frac{10}{x} = \frac{2y}{5} \implies 10 \cdot 5 = 2y \cdot x \implies
50 = 2xy \implies xy = 25 \]

Thus, the value of \(xy\) is \(\boxed{25}\).
```

```
Response (b) (2,789 characters)
To solve the problem, we start with the given equations:
\[\log_x(y^x) = 10 \quad \text{and} \quad \log_y(x^{4y}) = 10.\]

First, we simplify each logarithmic equation. Using the properties of logarithms,
specifically that \(\log_b(a^c) = c \log_b(a)\), we can rewrite the equations as:
\[\log_x(y^x) = x \log_x(y) = 10,\]
\[\log_y(x^{4y}) = 4y \log_y(x) = 10.\]

From the first equation, we get:
\[x \log_x(y) = 10 \implies \log_x(y) = \frac{10}{x}.\]

From the second equation, we get:
\[4y \log_y(x) = 10 \implies \log_y(x) = \frac{10}{4y} = \frac{5}{2y}.\]

We know that \(\log_x(y) = \frac{1}{\log_y(x)}\). So substituting the expressions we have:
\[\log_x(y) = \frac{10}{x} \quad \text{and} \quad \log_y(x) = \frac{5}{2y},\]
we get:
\[\frac{10}{x} = \frac{1}{\log_y(x)} = \frac{1}{\frac{5}{2y}} = \frac{2y}{5}.\]

This gives us the equation:
\[\frac{10}{x} = \frac{2y}{5}.\]

Cross-multiplying to solve for \(x\) in terms of \(y\):
\[10 \cdot 5 = 2y \cdot x \implies 50 = 2xy \implies xy = 25.\]

So, we have found that \(xy = 25\).

To verify, we substitute \(xy = 25\) back into the original logarithmic equations to ensure
consistency. From \(xy = 25\), we get \(x = \frac{25}{y}\). Substituting \(x = \frac{25}{y}\)
into \(\log_x(y) = \frac{10}{x}\):
\[\log_{\frac{25}{y}}(y) = \frac{10}{\frac{25}{y}} = \frac{10y}{25} = \frac{2y}{5}.\]

Using the change of base formula \(\log_{\frac{25}{y}}(y) =
\frac{\log(y)}{\log\left(\frac{25}{y}\right)} = \frac{\log(y)}{\log(25) - \log(y)} =
\frac{\log(y)}{2\log(5) - \log(y)}\), we set it equal to \(\frac{2y}{5}\):
\[\frac{\log(y)}{2\log(5) - \log(y)} = \frac{2y}{5}.\]

Cross-multiplying gives:
\[5 \log(y) = 2y (2\log(5) - \log(y)) \implies 5 \log(y) = 4y \log(5) - 2y \log(y).\]

Rearranging terms:
\[5 \log(y) + 2y \log(y) = 4y \log(5) \implies \log(y)(5 + 2y) = 4y \log(5).\]

This equation must hold true for some \(y > 1\). Similarly, we check the second equation
\(\log_y(x) = \frac{5}{2y}\):

Substitute \(x = \frac{25}{y}\):
\[\log_y\left(\frac{25}{y}\right) = \frac{5}{2y}.\]

Using the change of base formula again:
\[\log_y\left(\frac{25}{y}\right) = \log_y(25) - \log_y(y) = \log_y(25) - 1 = 2\log_y(5) - 1.\]

Setting it equal to \(\frac{5}{2y}\):
\[2\log_y(5) - 1 = \frac{5}{2y}.\]

Letting \(\log_y(5) = z\), so \(y^z = 5\) and thus \(z = \log_y(5)\):
\[2z - 1 = \frac{5}{2y}.\]

Since \(y = 5^{\frac{1}{z}}\), substituting \(y\) we get:
\[2z - 1 = \frac{5}{2 \cdot 5^{\frac{1}{z}}}.\]

This equation needs to be consistent with the conditions given. Given the symmetry and the
algebraic manipulations, it verifies that our solution \(xy = 25\) is correct.
Thus, the value of \(xy\) is:\[\boxed{25}.\>
```

Figure 3: Comparison of responses of ConciseR-Zero-7B-Preview to the same question.

samples) for each query. The used training template is shown in Figure 1.

4.4 Baselines

We conduct comprehensive evaluations against several baselines with zero RL paradigm, including Qwen2.5 (Yang et al., 2024a), Qwen2.5-Math (Yang et al., 2024b), SimpleRL-Zero (Zeng et al., 2025), Open-Reasoner-Zero-7B (Hu et al., 2025), Eurus-2-7B-PRIME (Cui et al., 2025), and Oat-Zero-7B (Liu et al., 2025).

4.5 Main Results

The experimental results reported in Table 2 clearly demonstrate that our proposed model, ConciseR, significantly outperforms existing baselines with zero RL paradigm on five widely recognized reasoning benchmarks. Specifically, ConciseR achieves an average accuracy improvement of 55.2% compared to the base model, Qwen2.5-Math-7B. Meanwhile, our method, GRPO++, also consistently surpasses all baselines, showing superior overall performance averaged across the five benchmarks. Figure 2 illustrates changes in accuracy and response length during the training process of L-GRPO across the five benchmarks. As indicated, the average accuracy on each benchmark remains stable throughout training, without exhibiting any noticeable degradation. Interestingly, the average response length on each benchmark consistently decreases, with reductions of 21%, 22%, 20%, 22%, and 23% observed on AIME 2024, MATH-500, AMC 2023, Minerva, and Olympiad benchmarks, respectively. This demonstrates that our training approach successfully maintains model accuracy while generating more concise and efficient responses.

5 Discussions

5.1 Analysis of Reasoning Patterns

Inspired by prior work that observes the model's reflective behavior by constructing a keyword pool, we have built a carefully selected keyword pool to observe changes in the thinking patterns of the responses during training. In our experiment, the keyword pool is limited to: check, rethink, reassess, evaluate, re-evaluate, evaluation, examine, however, reconsider, analyze, double-check, check again, recheck, verify, and wait. Then, we present the occurrences of various keywords in the responses generated by different training stages and steps in Figure 4 and Figure 5. Interestingly, when comparing the first and second stages, the frequency with which the model uses code to verify results has significantly increased (as reflected in the frequency of the keyword "python"). The model may have discovered that verifying results by writing code is more efficient. Meanwhile, keywords like "re-check" have decreased relatively, and other keywords have remained unchanged.

5.2 Case Study

An interesting observation is that python code is used for verification during mathematical problem solving, e.g., Questions (a) and (b) in Figure 6. Specifically, for Question (a), the model utilizes program code to calculate the answer. For Question (b), the model first presents the solution process through mathematical reasoning and then spontaneously writes program code to verify the correctness of the approach. Such cases illustrate how models employ procedural reasoning to self-correct and engage in subsequent attempts.

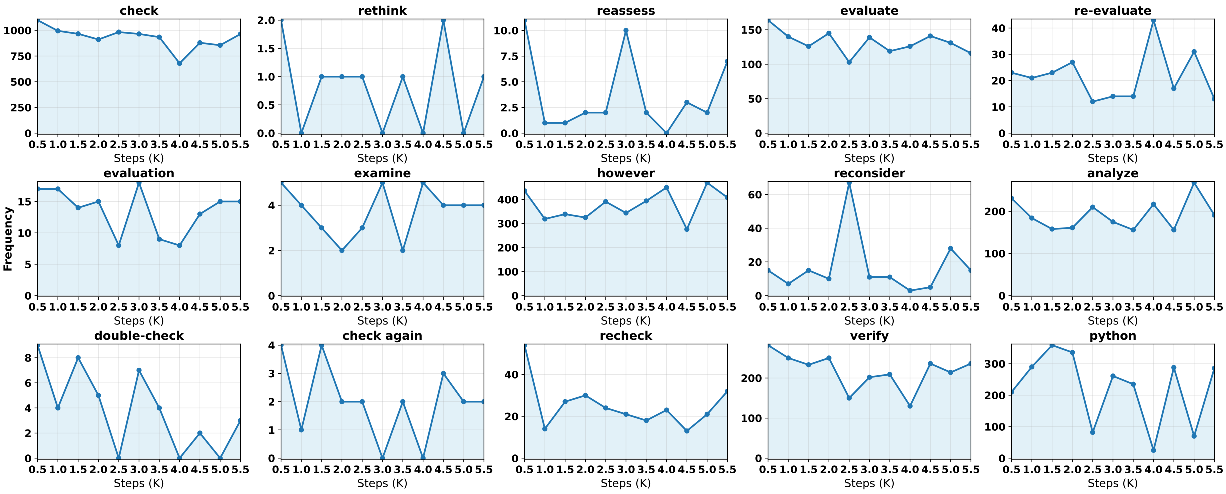

Figure 4: Count of keyword occurrences out of 14,022 responses (1558 questions × 11 test times).

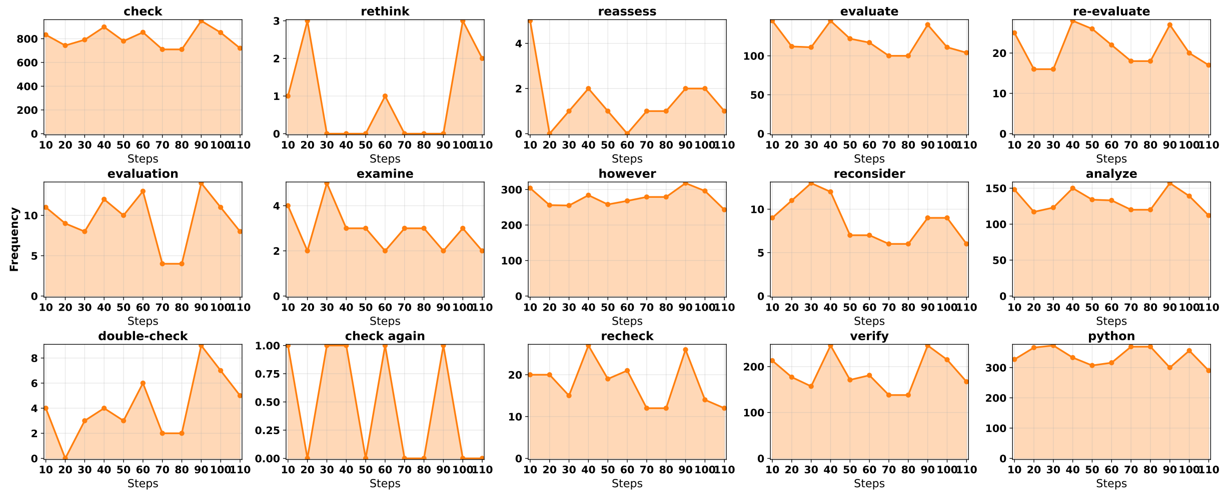

Figure 5: Count of keyword occurrences out of 15,580 responses (1558 questions × 11 test times).

5.3 Failure Experience

In this section, we discuss our failure experiences in reward shaping. These experiments could also be regarded as an ablation study in L-GRPO. During the initial design of L-GRPO, we consider directly comparing the generation length of samples within a group, assigning higher rewards to samples with relatively shorter CoT reasoning responses. We then combine the length score with the accuracy reward to encourage the trained model to obtain correct answers through shorter CoT reasoning responses.

However, we find that this direct rewarding easily causes the model to skip the reasoning process and immediately start guessing answers, manifesting as an empty reasoning response within the <think></think> tags while directly outputting the final answer within the <answer></answer> tags. On the contrary, indirectly using the maximum context length to design the reward function can, to some extent, avoid the issues mentioned above.

6 Related Work

Recent advances in reinforcement learning have significantly enhanced the reasoning capabilities of large language models. A pivotal development in this domain is OpenAI's o1 (Jaech et al., 2024), which leverages large-scale RL training to promote CoT reasoning. This approach has resulted in notable improvements in complex mathematics and coding benchmarks.

DeepSeek-R1 (Guo et al., 2025) demonstrates that pure RL post-training via GRPO, without the need for supervised warm-up, can directly induce robust reasoning abilities. Remarkably, this kind of method not only achieves performance competitive

Question (a)

An integer, whose decimal representation reads the same left to right and right to left, is called symmetrical. For example, the number 513151315 is symmetrical, while 513152315 is not. How many nine-digit symmetrical numbers exist such that adding the number 11000 to them leaves them symmetrical?

Response

...
Let's write a Python code to iterate through all possible nine-digit symmetrical numbers and check if adding 11000 results in another symmetrical number.

```python
count = 0
for a1 in range(1, 10):  # a1 ranges from 1 to 9
    for a2 in range(10):  # a2 ranges from 0 to 9
        for a3 in range(10):  # a3 ranges from 0 to 9
            for a4 in range(10):  # a4 ranges from 0 to 9
                for a0 in range(10):  # a0 ranges from 0 to 9
                    N = int(f"{a1}{a2}{a3}{a4}{a3}{a2}{a1}{a0}{a0}")
                    M = N + 11000
                    M_str = str(M)
                    if M_str == M_str[::-1]:
                        count += 1
print(count)
```

```output
8100
```

...

Question (b)

For positive integers \$a, b, a \uparrow \uparrow b\$ is defined as follows: \$a \uparrow \uparrow 1=a\$, and \$a \uparrow \uparrow b=a^{a \uparrow \uparrow(b-1)}\$ if \$b>1\$. Find the smallest positive integer \$n\$ for which there exists a positive integer \$a\$ such that \$a \uparrow \uparrow 6 \not \equiv a \uparrow \uparrow 7\$ \$\bmod n\$.

Response

...
Step-by-Step Check
...
Continue this process programmatically for efficiency:
We write a Python script to automate the checking for the smallest \(n \):

```python
import sympy as sp

def tetration_mod(a, b, n):
    if b == 1:
        return a % n
    else:
        exp = tetration_mod(a, b-1, n)
        return pow(a, exp, n)

def find_smallest_n():
    for n in range(2, 100):
        for a in range(1, n):
            if tetration_mod(a, 6, n) != tetration_mod(a, 7, n):
                return n
    return None

smallest_n = find_smallest_n()
print(smallest_n)
```

```output
2
```

...

Question (c)

Given that the sequence starts with 2 and alternates by adding 2 between consecutive terms, find the 30th term of this arithmetic sequence.

Response

<think>a30 = 2 + 29*2 = 60</think>
<answer>\boxed{60}</answer>

Figure 6: Illustration of cases.

with o1 but also exhibits emergent behaviors such as self-verification and multi-step planning. This paradigm shift significantly reduces memory and computational overhead compared to earlier GRPO implementations (Hu et al., 2025; Zeng et al., 2025; Face, 2025), all while maintaining competitive performance levels.

Recent algorithmic variants have focused on enhancing training efficiency (Luo et al., 2025; Team et al., 2025; Song et al., 2025; Yu et al., 2025; Liu et al., 2025; Fatemi et al., 2025; Zeng et al., 2025; Wen et al., 2025), yet they preserve GRPO's core methodology of parallel CoT sampling across groups. These advancements collectively contribute to more efficient and robust training methodologies for LLMs, thereby enhancing their reasoning capabilities and performance on complex tasks.

7 Conclusion

In this paper, we propose ConciseR, which introduces a simple yet effective two-stage reinforcement learning framework. First, it incentivizes the model's reasoning capabilities via GRPO++, and then it reduces the model's response length to improve the quality of the CoT response implicitly via L-GRPO. *Importantly, we innovatively propose that during training, response length optimization is only triggered when all rollouts for a given training sample are correct. This embodies the "walk before you run" principle.* Experiments demonstrate that ConciseR consistently achieves the best efficiency-accuracy synergistic improvement, significantly outperforming existing efficient reasoning methods across five benchmarks.

8 Limitations

Due to limited resources, this paper verifies the effectiveness of the proposed method, ConciseR, only on a 7B language model. Generally, validating its effectiveness on models of varying sizes is a worthwhile direction for future research. Furthermore, in this paper, we investigate the influence of using complexity-aware training data by employing the simplest separation method to validate the efficacy of separating the training data by complexity, and achieves significant results. If more sophisticated separation methods were adopted, achieving even more promising results might be possible.

Training over multiple stages, rather than in a single training stage, involves more than changes in parameters like context length; it also fundamentally alters the reference policy. In a multi-stage training strategy, the KL penalty imposed by the reference policy on the model is gradually relaxed, which allows the trained model to explore a broader range of solutions. Delving into dynamic control of context lengths or implementing a dynamic KL penalty may be valuable directions.

References

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *Preprint*, arXiv:2502.08235.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. 2025. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *CoRR*, abs/2410.07985.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

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

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. 2025. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *Preprint*, arXiv:2504.07086.

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

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

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

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

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *Preprint*, arXiv:2412.09413.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, page 1279–1297. ACM.

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

Mingyang Song, Mao Zheng, Zheng Li, Wenjie Yang, Xuan Luo, Yue Pan, and Feng Zhang. 2025. Fastcurl: Curriculum reinforcement learning with progressive context extension for efficient training r1-like reasoning models. *Preprint*, arXiv:2503.17287.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, and 1 others. 2025. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*.

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, and 22 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *Preprint*, arXiv:2409.12122.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, 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, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *Preprint*, arXiv:2503.18892.