

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 RECTIFYING LLM THOUGHT FROM LENS OF OPTIMIZATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Recent advancements in large language models (LLMs) have been driven by their emergent reasoning capabilities, particularly through long chain-of-thought (CoT) prompting, which enables thorough exploration and deliberation. Despite these advances, long-CoT LLMs often exhibit suboptimal reasoning behaviors, such as overthinking and excessively protracted reasoning chains, which can impair performance. In this paper, we analyze reasoning processes through an optimization lens, framing CoT as a gradient descent procedure where each reasoning step constitutes an update toward problem resolution. Building on this perspective, we introduce REPRO (Rectifying Process-level Reward), a novel approach to refine LLM reasoning during post-training. REPRO defines a surrogate objective function to assess the optimization process underlying CoT, utilizing a dual scoring mechanism to quantify its intensity and stability. These scores are aggregated into a composite process-level reward, seamlessly integrated into reinforcement learning with verifiable rewards (RLVR) pipelines to optimize LLMs. Extensive experiments across multiple reinforcement learning algorithms and diverse LLMs, evaluated on benchmarks spanning mathematics, science, and coding, demonstrate that REPRO consistently enhances reasoning performance and mitigates suboptimal reasoning behaviors.

1 INTRODUCTION

Recent advancements in large language models (LLMs) have been propelled by their emergent reasoning capabilities, enabling them to tackle complex tasks (Huang & Chang, 2023; Plaat et al., 2024; Ahn et al., 2024; Ke et al., 2025; Sun et al., 2025). These capabilities are pivotal in progressing toward artificial general intelligence (AGI) (Zhong et al., 2024). State-of-the-art LLMs, such as OpenAI’s o-series (OpenAI, 2024a,b; 2025), DeepSeek-R1 (DeepSeek-AI et al., 2025), Kimi-K1 (Kimi-Team et al., 2025), and Gemini-2.5-Pro (Comanici et al., 2025), leverage long chain-of-thought (CoT) prompting to enhance reasoning. This approach facilitates comprehensive exploration and reflection, yielding robust reasoning processes (Chen et al., 2025a). Such improvements stem largely from reinforcement learning with verifiable rewards (RLVR) (Schulman et al., 2017; Shao et al., 2024), which enables LLMs to autonomously explore reasoning steps based on a terminal reward, fostering self-improving models with scalable reasoning during inference (Snell et al., 2024).

Despite these advancements, long-CoT LLMs often exhibit suboptimal reasoning behaviors (Chen et al., 2025a). A significant issue is overthinking, where models generate excessive tokens or protracted reasoning paths that contribute minimally to problem resolution, incurring substantial computational costs (Chen et al., 2024; Wang et al., 2025c; Sui et al., 2025). For instance, in response to a simple query like “What is the answer to 2 plus 3?” (Chen et al., 2024), certain long-CoT LLMs produce reasoning chains exceeding thousands of tokens, increasing latency and resource demands, thus limiting applicability in time-sensitive domains (Sui et al., 2025).

Drawing on prior work (Feng et al., 2023; Huang et al., 2025a), we analyze suboptimal reasoning through an optimization framework, conceptualizing CoT as a task-specific variant of gradient descent, where each reasoning step represents an optimization update (Liu et al., 2025a). In this paradigm, suboptimal reasoning manifests as oscillations around saddle points or local optima, hindering convergence to the optimal solution.

To address these challenges, we propose REPRO (**R**ectifying **P**rocess-level **R\mathcal{J}, to monitor the optimization process of CoT, measuring the LLM’s confidence in the ground truth via perplexity (Jelinek et al., 1977) over the ground-truth token sequence. For a reasoning trajectory of N steps, we compute a sequence of objective values $[\mathcal{J}_0, \mathcal{J}_1, \dots, \mathcal{J}_N]$ and introduce a dual scoring system to assess optimization intensity and stability. These scores are combined into a composite process-level reward (Lightman et al., 2024), integrated into standard post-training pipelines (DeepSeek-AI et al., 2025; Shao et al., 2024; Hu, 2025) to enhance reasoning. REPRO is plug-and-play, compatible with prevalent reinforcement learning algorithms.**

The efficacy of REPRO is substantiated by comprehensive empirical evaluation. We validate REPRO through extensive experiments using reinforcement learning algorithms like PPO (Schulman et al., 2017), REINFORCE++ (Hu, 2025), REINFORCE++ Baseline (Hu, 2025), and GRPO (Shao et al., 2024), across LLMs of various families and scales, including base models, supervised fine-tuned variants, and native long-CoT LLMs. Evaluations on benchmarks in mathematics, science, and coding demonstrate significant improvements in reasoning performance. Quantitative and qualitative analyses further confirm REPRO’s efficacy in optimizing reasoning behaviors. Our contributions are: ① We introduce REPRO, a plug-and-play method to rectify LLM reasoning in RLVR; ② We define a surrogate objective function to model reasoning as gradient descent, with a dual scoring mechanism for optimization intensity and stability, and outline its integration as a process-level reward; ③ Extensive experiments across reinforcement learning algorithms and LLMs show enhanced reasoning performance; ④ Quantitative and qualitative analyses verify REPRO’s ability to refine LLM reasoning behaviors.

2 PRELIMINARIES

Reinforcement Learning for LLM Reasoning. Proximal Policy Optimization (PPO) (Schulman et al., 2017) is the typical and effective policy gradient algorithm for LLM post-training (Ouyang et al., 2022; Hu et al., 2025). As an actor-critic method, PPO employs a policy model (*actor*) to optimize a reward function and a value model (*critic*) to estimate the value of each state. PPO employs the clipped surrogate objective function to enhance training stability by constraining the magnitude of policy updates at each iteration with a clipping range ϵ . Given the input data distribution P and policy model π_θ , the objective is formally defined as:

$$\mathcal{J}(\theta) = \mathbb{E}_{q \sim P, \tau \sim \pi_\theta} \left[\frac{1}{|\tau|} \sum_{t=1}^{|\tau|} \left\{ \min(\rho_t A_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon) A_t) \right\} \right], \quad (1)$$

where $\rho_t = \pi_\theta(\tau_{(t)}|q, \tau_{(\leq t)}) / \pi_{\theta_{\text{old}}}(\tau_{(t)}|q, \tau_{(\leq t)})$ is the importance sampling coefficient to reduce the gap between the current policy and the old policy. A_t denotes the advantage estimate at time step t , which is computed using Generalized Advantage Estimation (GAE) (Schulman et al., 2016). GAE is derived from the temporal difference error, $\delta_t = r_t + \gamma V_{t+1} - V_t$, where r_t is the reward at time step t , γ is the discount factor, and V_t is the value at time step t . Then A_t is calculated by the summation of the temporal difference error over a series of time steps as: $A_t = \sum_{i=0}^{\infty} \gamma^i \delta_{t+i}$.

Critic-Free RL Algorithms for LLM Reasoning. Despite the effectiveness of PPO, it experiences high computational costs due to the trainable value model. To address this challenge, a series of critic-free RL algorithms have been proposed, substituting the value V_t with an estimated reward baseline. These include ReMax (Li et al., 2024), RLOO (Ahmadian et al., 2024), GRPO (DeepSeek-AI et al., 2025; Shao et al., 2024), and REINFORCE++ (Hu, 2025). Typically, these algorithms share the following objective function:

$$\mathcal{J}(\theta) = \mathbb{E}_{q \sim P, \{\tau_i\} \sim \pi_\theta} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|\tau_i|} \sum_{t=1}^{|\tau_i|} \left\{ \min(\rho_{i,t} \tilde{A}_t^i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon) \tilde{A}_t^i) - \beta D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}] \right\} \right], \quad (2)$$

where $\tau_i = \{\tau_1, \dots, \tau_G\} \sim \pi_{\theta_{\text{old}}}(\cdot|q)$ denotes a group of trajectories of size G generated by the existing policy model π_θ . \tilde{A}_t^i represents the normalized advantage using an estimated reward baseline at time step t for the i -th trajectory. $D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]$ denotes the KL divergence penalty between the current policy π_θ and the reference policy π_{ref} , with β as the weighting factor for this penalty term.

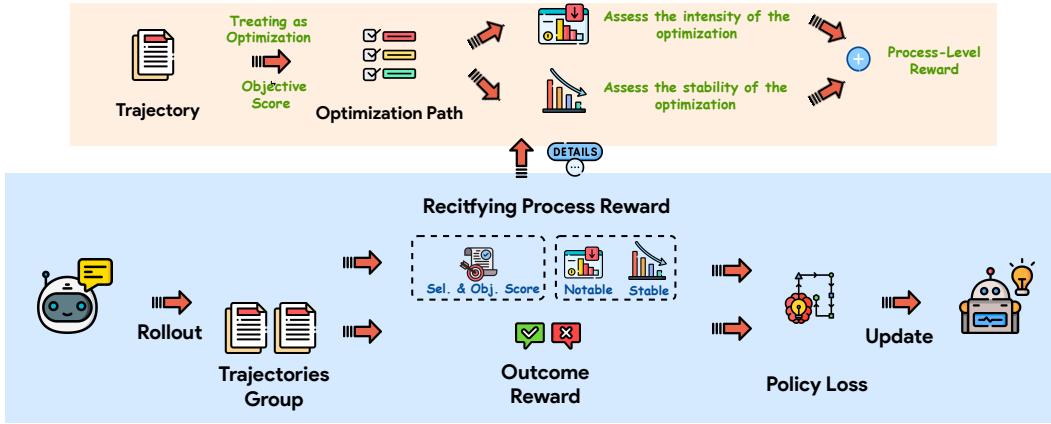


Figure 1: Illustration of the REPRO framework. We incorporate a rectifying process-level reward into the RLVR training to enhance LLM reasoning. Initially, we conceptualize the reasoning trajectories generated by LLMs as an optimization process of the LLMs’ internal state (§ 3.1 & § 3.2). We then propose a two-fold score to evaluate the optimization process and utilize this score as a reward to rectify the LLM thought (§ 3.3 & § 3.4).

3 REPRO: RECTIFYING LLM THOUGHT

In this section, we provide the details of the proposed REPRO and the illustration of REPRO is demonstrated in Figure 1.

3.1 PROBLEM FORMULATION

A typical LLM reasoning process involves a question q randomly sampled from the question distribution $P(Q)$, denoted as $q \sim P(Q)$, and an LLM parameterized by π_θ . For q , a long-CoT LLM generates a step-by-step reasoning sequence τ_{thinking} (typically delimited by `<think>` and `</think>` tags in current reasoning LLMs), followed by a conclusion $\tau_{\text{conclusion}}$, forming the trajectory:

$$\tau = [\tau_{\text{thinking}}; \tau_{\text{conclusion}}] \sim \pi_\theta(\cdot | q). \quad (3)$$

Following prior work (Liu et al., 2025a; Wang et al., 2025a), we conceptualize the decoding of τ_{thinking} as an optimization process over the LLM’s internal states, iteratively increasing the likelihood of the correct answer. The objective function $\mathcal{J}(\pi_\theta, q, \tau, a)$, where a is the ground-truth answer, is optimized as:

$$\theta_{t+1} \leftarrow \theta_t + \tilde{\eta} \cdot \tilde{\nabla}_\theta \mathcal{J}(\pi_\theta, q, \tau_{(\leq t)}, a), \quad \theta^* = \arg \max_{\theta} \mathcal{J}(\pi_\theta, q, \tau, a), \quad (4)$$

where $\tilde{\eta}$ is an implicit learning rate, and $\tilde{\nabla}_\theta \mathcal{J}(\pi_\theta, q, \tau_{(\leq t)}, a)$ denotes the implicit gradient of $\mathcal{J}(\pi_\theta, q, \tau_{(\leq t)}, a)$ with respect to θ , as the actual optimization process is complex and nontrivial.

3.2 OBJECTIVE FUNCTION DEFINITION

Although the actual optimization process is complex and nontrivial, we can define a proxy metric to observe changes in the objective function from an indirect perspective. Drawing inspiration from previous work (Tang et al., 2025; Zhou et al., 2025; Yu et al., 2025b), we find that the probability of the model generating the ground truth answer a serves as an effective proxy for the objective function \mathcal{J} . Formally, we define the proxy objective function as follows:

$$\tilde{\mathcal{J}}(\pi_\theta, q, \tau_{(\leq t)}, a) \triangleq \frac{1}{|a|} \sum_{i=1}^{|a|} \log \pi_\theta(a_{(i)} | q, \tau_{(\leq t)}). \quad (5)$$

Intuitively, $\tilde{\mathcal{J}}$ quantifies the model’s reasoning capability given certain context. As $\tau_{\leq t}$ updates the model’s internal states, the probability of producing the ground-truth answer increases, thereby increasing $\tilde{\mathcal{J}}$.

162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

Empirical Evidence. We provide empirical evidence supporting the effectiveness of $\tilde{\mathcal{J}}$ as a proxy metric. Specifically, we prompt LRM (DeepSeek-R1-Distill-Qwen-1.5B) (DeepSeek-AI et al., 2025) with a mathematical question sampled from AIME’24¹ to generate multiple reasoning trajectories. We select four correct trajectories, computing $\tilde{\mathcal{J}}$ (to demonstrate, we show the negative value of $\tilde{\mathcal{J}}$) at each position of the trajectory, and plot the curve as shown in Figure 2. From Figure 2, we observe that $-\tilde{\mathcal{J}}$ gradually decreases as the reasoning trajectory length increases. This indicates that $\tilde{\mathcal{J}}$ effectively serves as a proxy metric for monitoring and assessing the internal states of the LLM.

3.3 QUANTIFYING OPTIMIZATION PROCESS

Leveraging the proposed objective function $\tilde{\mathcal{J}}$, we introduce a score \mathcal{S} designed to evaluate the optimization process by tracking the dynamics of $\tilde{\mathcal{J}}$. For a given reasoning trajectory τ , a sequence of $\tilde{\mathcal{J}}$ values, denoted as $\{\tilde{\mathcal{J}}_1, \tilde{\mathcal{J}}_2, \dots, \tilde{\mathcal{J}}_{|\tau|}\}$, is obtained, which represents the optimization process over the generation of τ . An effective optimization process should fulfill two key conditions: 1) the value of objective function exhibits a sufficient overall increase, indicating substantial progress to the optimization objective; 2) the increase is relatively smooth, with limited oscillation near local extrema, indicating efficient optimization. Building on these criteria, we propose a dual quantitative score, \mathcal{S} , to evaluate the optimization process. This score comprises the **Magnitude Score**, $\mathcal{S}_{\text{magn}}$ measuring the *intensity* of the optimization process (i.e., net improvement), and the **Stability Score**, $\mathcal{S}_{\text{stab}}$, assessing its *stability*, capturing the degree of oscillatory behavior in the updates.

Magnitude Score. The magnitude score, $\mathcal{S}_{\text{magn}}$, at position t (denoted as $\mathcal{S}_{\text{magn},(t)}$), quantifies the increase in $\tilde{\mathcal{J}}$ along the partial trajectory $\tau_{\leq t}$. To address the disparities among $\tilde{\mathcal{J}}$ values corresponding to different \mathbf{q} , a baseline $\bar{\mathcal{J}}_b(\mathbf{q})$ is introduced, defined as:

$$\bar{\mathcal{J}}_b(\mathbf{q}) \triangleq \tilde{\mathcal{J}}(\pi_\theta, \mathbf{q}, \mathbf{a}), \quad (6)$$

which can be interpreted as the direct probabilistic prediction from π_θ . Subsequently, $\mathcal{S}_{\text{magn},(t)}$ is defined as:

$$\mathcal{S}_{\text{magn},(t)} \triangleq \tanh(\Delta(\pi_\theta, \mathbf{q}, \tau_{\leq t}, \mathbf{a}) + 1) + 1 \in (0, 1],$$

$$\text{where } \Delta(\pi_\theta, \mathbf{q}, \tau_{\leq t}, \mathbf{a}) = \frac{\tilde{\mathcal{J}}(\pi_\theta, \mathbf{q}, \tau_{\leq t}, \mathbf{a}) - \bar{\mathcal{J}}_b(\mathbf{q})}{\bar{\mathcal{J}}_b(\mathbf{q})} \quad (7)$$

Intuitively, $\mathcal{S}_{\text{magn},(t)}$ is a normalized measure of the relative increase of $\tilde{\mathcal{J}}$ over the baseline $\bar{\mathcal{J}}_b$. Normalization of this relative decrease $\Delta(\pi_\theta, \mathbf{q}, \tau_{\leq t}, \mathbf{a})$ through the tanh function ensures the score’s range is restricted to $(0, 1]$, thus mitigating the impact of extreme values. A higher $\mathcal{S}_{\text{magn},(t)}$ signifies a greater increase in the objective function, concurrently indicating that the partial reasoning trajectory $\tau_{\leq t}$ yields more substantial benefits to the reasoning process.

Stability Score. As previously stated, the $\mathcal{S}_{\text{stab}}$ quantifies the stability of the optimization process. Each step is expected to serve as an effective update, progressing towards increasing the objective function. For a given sequence of objective values $\{\tilde{\mathcal{J}}_1, \tilde{\mathcal{J}}_2, \dots, \tilde{\mathcal{J}}_{|\tau_{\leq t}|}\}$, we evaluate $\mathcal{S}_{\text{stab},(t)}$ by examining its correlation with the corresponding indices $\{1, 2, \dots, |\tau_{\leq t}|\}$, as follows:

$$\mathcal{S}_{\text{stab},(t)} = \frac{\sum_{i < j, 1 \leq i, j \leq |\tau_{\leq t}|} \text{sign}(\tilde{\mathcal{J}}_i - \tilde{\mathcal{J}}_j) \cdot \text{sign}(i - j)}{|\tau_{\leq t}|(|\tau_{\leq t}| - 1)} + \frac{1}{2} \in [0, 1], \quad (8)$$

¹https://huggingface.co/datasets/HuggingFaceH4/aime_2024

where sign is the sign function. Theoretically, we leverage Kendall’s Tau Correlation Coefficient (Kendall, 1938) to measure the stability of the optimization process. If each step is an effective update, S_{stab} approaches 1, whereas ineffective updates result in a score near 0. In addition, considering the influence of noise, smooth can also be introduced to smooth \tilde{J}_i , such as the common EMA (Exponential Moving Average) (Hunter, 1986) smoothing equation:

$$\tilde{J}_{i,t} = \alpha \cdot \tilde{J}_{i,t-1} + (1 - \alpha) \cdot \tilde{J}_{i,t}, \quad (9)$$

where α is the smoothing factor.

Finally, combining S_{magn} and S_{stab} , we obtain the final score S by introducing a weight factor $w \in [0, 1]$ as follows:

$$S = (1 - w) \cdot S_{\text{magn}} + w \cdot S_{\text{stab}}. \quad (10)$$

3.4 LEARNING WITH RECTIFYING PROCESS-LEVEL REWARD

We shall now discuss the integration of S into the RL training for LLMs. A natural approach is to employ S as a process reward in reinforcement learning training. Nonetheless, since current strong reasoning LLMs generate lengthy reasoning trajectories, computing S at each token would incur prohibitive computational overhead. Moreover, token-level S calculation could introduce excessive noise, leading to futile computations of S values that adversely affect the optimization process.

Entropy-Based Selection Strategy. Recent studies (Cui et al., 2025; Wang et al., 2025b; Cheng et al., 2025) have demonstrated that token entropy serves as an effective indicator within the trajectories of advanced reasoning LLMs. High-entropy tokens act as critical decision points, guiding the model toward diverse reasoning pathways. In this study, where reasoning trajectory generation is framed as an optimization process, high-entropy tokens may cause oscillations near extrema, yielding higher value than tokens with lower entropy. Therefore, to reduce computational overhead while providing more effective process rewards, we propose an entropy-based selection strategy. Specifically, we divide the thinking tokens within the reasoning trajectory into multiple segments (e.g., partitioned by two-line-break `\n\n`), considering the thinking granularity of LLMs:

$$\tau_{\text{thinking}} \mapsto \{c_1, c_2, \dots, c_N\}, \quad (11)$$

where N denotes the number of segments. We select the top- k segments $\{\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_k\}$ based on the entropy of the first token of each segment:

$$\text{top-}k(\mathcal{H}(c_{1,(0)}), \mathcal{H}(c_{2,(0)}), \dots, \mathcal{H}(c_{N,(0)})) \mapsto \{\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_k\}. \quad (12)$$

The rationale for this strategy is that model uncertainty increases at the conclusion of these segments, which, from an optimization perspective, indicates that suboptimal optimization processes are more likely to occur, a phenomenon we seek to rectify.

Integrating S Into Reward. Given the selected segments $\{\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_k\}$, we compute scores $\{S_1, S_2, \dots, S_k\}$ for each segment by appending the ground truth answer at the end. The rectifying process-level reward \tilde{r}_j is computed as:

$$\tilde{r}_j = \begin{cases} S_j - S_{j-1} & \text{if } j > 1, \\ S_j & \text{if } j = 1, \end{cases} \quad (13)$$

which signifies the gain from introducing the partial trajectory from the end of \tilde{c}_{j-1} to \tilde{c}_j . \tilde{r}_j rectifies the thinking process of LLMs by penalizing thinking processes associated with suboptimal optimization while encouraging those associated with optimal optimization. Subsequently, we also involve the normalization in critic-free RL algorithms on the rectifying process-level reward \tilde{r}_t to mitigate the probabilistic prediction mismatch between different q and facilitate stable policy updates:

$$\tilde{r}'_j = \text{Norm}(\tilde{r}_j | \{\tilde{r}_{j,i}\}_i), \quad (14)$$

where $\{\tilde{r}_{j,i}\}_i$ denotes the specific group for the normalization of r'_j and refer to § C.1 for more details. We separate the normalization of process-level reward from outcome reward to prevent interference with the correctness reward from noise signals. For a token $\tau_{(t)}$ where $\text{index}(c_{j-1,(\lfloor c_{j-1} \rfloor)}) < t \leq \text{index}(c_{j,(0)})$ and $\text{index}(x)$ denotes the index of token x , the rectifying process-level advantage \tilde{A}_t and overall advantage \hat{A}_t are defined as $\tilde{A}_t = \sum_{i=j}^k \tilde{r}'_i$, $\hat{A}_t = A + \alpha \cdot \tilde{A}_t$ where A is the verifiable

270 outcome advantage to measure the correctness of the overall trajectory, computed from the correctness
 271 of reasoning trajectories, and α is the weight factor.
 272

273 By synthesizing all propositions and definitions outlined above, the LLM parameters are updated
 274 using the computed advantage \hat{A}_t following the clipped surrogate objective specified in Equation (2).
 275

Empirical Example For Understanding \tilde{r} . To illustrate the effectiveness of the rectifying process-level reward \tilde{r} , we present an empirical example. As shown in Example 3.1, we prompt the LRM to answer a question, compute \tilde{r} for each reasoning step, and identify steps with low rectifying process-level rewards. We find that steps with low \tilde{r} typically involve self-doubt or redundant re-examination, contributing little to the reasoning process and increasing inefficiency. Conversely, steps with high \tilde{r} correspond to critical computational actions or preliminary conclusions that significantly advance reasoning. Thus, we propose that penalizing low- \tilde{r} steps and incentivizing high- \tilde{r} steps can optimize the model’s reasoning behavior.
 283

284 **Example 3.1: Empirical Example for Rectifying Process-Level Reward**

285 **QUESTION:**

286 There exist real numbers x and y , both greater than 1, such that
 287

$$288 \log_x(y^x) = \log_y(x^{4y}) = 10.$$

289 Find xy .
 290

291 **SELECTED REASONING PROCESS WITH HIGH/LOW RECTIFYING PROCESS-LEVEL
 292 REWARD:**

293 *($\tilde{r} = 0.143$) Alright, so I have this problem here: There exist real numbers x and y , both greater
 294 than 1. ... ($\tilde{r} = -0.217$) Hmm, okay. Let’s see. I remember that logarithms can be tricky, but
 295 maybe I can use some logarithmic identities to simplify these equations. I think the change of base
 296 formula might be helpful here. The change of base formula says that $\log_b a = \frac{\ln a}{\ln b}$, right? So
 297 maybe I can rewrite both equations using natural logarithms to make them easier to handle. ...
 298 ($\tilde{r} = -0.053$) That seems a bit cleaner. So, let me write those. ... ($\tilde{r} = -0.060$) This trial and error
 299 method isn’t working. Maybe I should consider a different approach. ... ($\tilde{r} = 0.085$) Left side:
 300 $1(1 + 10e^{-1}) \approx 1 + 10 \times 0.3679 \approx 1 + 3.679 \approx 4.679$, which is greater than 3.2188. ... ($\tilde{r} =$
 301 -0.113) Wait, that can’t be. Wait, perhaps my approximation is off. ... Wait, perhaps my approach is
 302 not efficient. ($\tilde{r} = -0.086$) Maybe I can use linear approximation or try to set up an equation. ... ($\tilde{r} =$
 303 0.053) Which is the same equation as before. So, this equation is satisfied when $xy = 25$. So, $xy = 25$
 304 is the solution. ... ($\tilde{r} = 0.092$) Therefore, despite the complexity of the original logarithmic equations,
 305 the product xy simplifies directly to 25.*

306
 307 **4 EXPERIMENTAL RESULTS AND ANALYSIS**

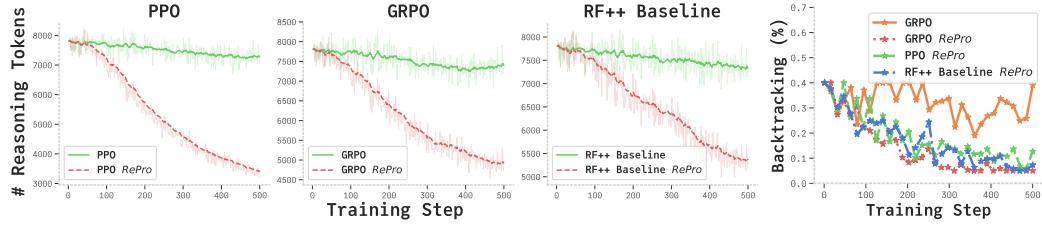
308 **4.1 SETUP**

311 **Evaluation Benchmarks.** We evaluate all models across three domain benchmarks: 1) *Mathematical*
 312 *Reasoning Benchmarks*, which include AIME24, AIME25, and MATH500 (Hendrycks et al., 2021b),
 313 and LiveMathBench (Liu et al., 2024); 2) *Scientific Reasoning Benchmarks*, represented by GPQA-
 314 Diamond (Rein et al., 2023); 3) *Code Reasoning Benchmarks*, which comprise MBPP (Austin et al.,
 315 2021) and LiveCodeBench (Jain et al., 2025).

316 **Implementation Details.** We conduct experiments on several prominent LLMs, including DeepSeek-
 317 R1-Distill-Qwen-1.5B (DeepSeek-AI et al., 2025), distilled from DeepSeek-R1 (DeepSeek-AI et al.,
 318 2025), Qwen3-1.7B (Yang et al., 2025), Qwen3-8B (Yang et al., 2025), Hunyuan-1.8B-Instruct (Ten-
 319 cent, 2025), and MobileLLM-R1-950M (Liu et al., 2025c). The training corpus, proposed by Luo
 320 et al. (2025), comprises approximately 40,000 high-quality mathematical samples. More details of the
 321 model training are provided in § C.4. For evaluation, we configured the sampling temperature to 0.6,
 322 top- p to 0.95, and top- k to 40. It is worth mentioning that different LLMs have their recommended
 323 sampling parameters, and different sampling parameters may have a certain impact on performance.
 324 However, to ensure a consistent evaluation pipeline, we use same sampling parameters for all models

324 Table 1: Performance of REPRO on evaluation reasoning benchmarks. We report the average
 325 performance for 16 runs on AIME24 and AIME25, and 4 runs on others. We abbreviate LMB as
 326 LiveMathBench v202505, LCB as LiveCodeBench v6, RF++ as REINFORCE++, and RF++ B
 327 as REINFORCE++ Baseline. ♠ denotes the in-domain evaluation benchmark and ♣ denotes the
 328 out-of-domain benchmark.

330 Methods	AIME24 ♠ % Avg@16 ↑	AIME25 ♠ % Avg@16 ↑	MATH500 ♠ % Avg@4 ↑	LMB ♠ % Avg@4 ↑	GPQA-D ♣ % Avg@4 ↑	MBPP ♣ % Avg@4 ↑	LCB ♣ % Avg@4 ↑
332 DeepSeek-R1-Distill-Qwen-1.5B							
333 Original	30.6	24.8	84.4	10.5	32.7	61.8	14.9
334 PPO	34.8	24.4	86.9	14.0	32.1	61.0	17.0
335 +REPRO	36.3	27.7	87.7	16.5	32.8	61.1	16.7
336 RF++	31.0	23.5	85.4	11.5	33.1	61.0	15.1
337 +REPRO	33.1	26.7	86.1	12.0	34.3	61.9	16.1
338 RF++ B	33.1	26.7	86.1	12.0	34.3	61.9	16.1
339 +REPRO	35.6	26.5	87.2	15.6	35.4	63.9	17.3
340 GRPO	32.9	25.3	86.0	10.3	34.5	62.5	15.2
341 +REPRO	36.0	26.5	87.1	14.3	37.0	65.4	18.4
342 Qwen3-1.7B							
343 Original	46.8	36.1	93.0	18.8	39.5	66.9	30.6
344 GRPO	47.3	34.8	93.4	18.8	38.3	67.5	32.2
345 +REPRO	49.8	37.9	94.1	19.5	39.1	68.8	32.0



352 Figure 3: Dynamics of the reasoning token cost during the training
 353 process of REPRO on DeepSeek-R1-Distill-Qwen-1.5B.

354 Figure 4: Dynamics of the
 355 backtracking pattern.

356 and focus on the *relative* improvement in performance. Also, to minimize variance, we report average
 357 performance relative to the size of each benchmark.

358 4.2 EFFECTIVENESS AND GENERALIZATION OF REPRO

360 Table 1 illustrates the performance of REPRO on evaluation reasoning benchmarks. From the
 361 experimental results, we have the following findings.

362 **REPRO Achieves Consistent Improvements Across RL Algorithms.** As presented in Table 1, RE-
 363 PRO consistently enhances the performance of various RL baselines, including PPO, REINFORCE++,
 364 REINFORCE++ Baseline, and GRPO. For instance, when applied to PPO on the DeepSeek-R1-
 365 Distill-Qwen-1.5B backbone, REPRO improves AIME24 accuracy from 34.8 to 36.3 and MATH500
 366 from 86.9 to 87.7. These results demonstrate that REPRO’s improvements are independent of the RL
 367 algorithm, offering a versatile, plug-and-play strategy to enhance reasoning performance.

368 **REPRO Generalizes to Out-of-Domain Benchmarks.** Beyond in-domain reasoning tasks like
 369 AIME and MATH500, REPRO exhibits robust generalization to out-of-domain benchmarks, such as
 370 science reasoning benchmark GPQA-Diamond and code reasoning benchmarks including MBPP and
 371 LiveCodeBench. These findings underscore REPRO’s ability to extend benefits beyond mathematical
 372 reasoning to diverse reasoning tasks like science reasoning, programming, and code generation tasks,
 373 highlighting its broad applicability.

374 **REPRO Generalizes to Diverse LLMs.** REPRO’s effectiveness extends across different LLMs
 375 including DeepSeek-R1-Distill-Qwen-1.5B and Qwen3-1.7B. We also provide results of different
 376 architectures and sizes in §§ D.1 and D.2. REPRO achieves consistent improvements across LLMs of
 377 different families and sizes. This scalability across model architectures and sizes indicates that REPRO

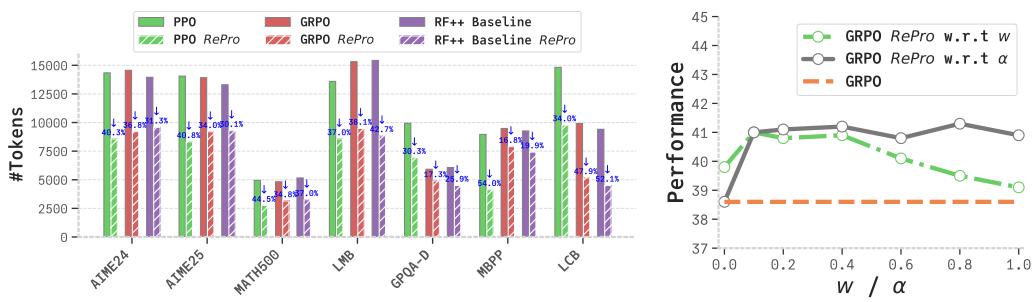


Figure 5: Comparison of the inference reasoning token cost of DeepSeek-R1-Distill-Qwen-1.5B. Figure 6: Ablation experiments of weight w and REPRO weight α .

Table 2: Ablation study of the number of selected segments k .

# k	AIME24 ♠	AIME25 ♠	MATH500 ♠	LMB ♠	GPQA-D ♣	MBPP ♣	LCB ♣
	% Avg@16 ↑	% Avg@16 ↑	% Avg@4 ↑				
DeepSeek-R1-Distill-Qwen-1.5B							
10	36.0	26.5	87.1	14.3	37.0	55.4	18.4
20	36.5	26.2	87.3	14.7	37.5	55.0	19.1
30	36.9	27.2	87.8	15.1	37.6	55.3	19.5

serves as a general mechanism for enhancing reasoning capabilities, rather than an optimization specific to a particular backbone.

4.3 ABLATION STUDY

Impact of Magnitude Score & Stability Score. Magnitude Score and Stability Score are utilized to assess the intensity and stability of the optimization process, respectively, with the coefficient w balancing their contributions. To investigate their necessity, we analyze model performance across different w values, as shown in Figure 6. The dotted lines, segmented by dots, represent the average performance of models trained with varying w values on four mathematical reasoning benchmarks. Across all w values, REPRO consistently outperforms the baseline, confirming the importance of both Magnitude Score and Stability Score. Notably, performance is slightly higher when w is lower, suggesting that the Magnitude Score, which measures optimization intensity, may play a more critical role in enhancing model performance.

Impact of the REPRO Weight α . To balance the outcome advantage and REPRO advantage, we introduce the coefficient α , set to 0.1. In this part, we evaluate the sensitivity of REPRO to variations in α . As depicted in Figure 6, the solid lines, segmented by dots, represent the average performance of models trained with different α values across four mathematical reasoning benchmarks. The results demonstrate that REPRO maintains relatively stable performance across various α values, indicating its robustness to changes in the balance coefficient.

Impact of the Number of Selected Segments k . The number of selected segments k is a key factor in balancing performance and training cost in REPRO. As shown in Table 2, while increasing k yields slight performance improvements, these gains are marginal. In practical applications, finding an optimal trade-off between training cost and performance is crucial. Additional experimental results and analysis are provided in § D.4.

4.4 FURTHER ANALYSIS OF THE CHANGING OF THINKING BEHAVIORS

In this section, we will conduct quantitative and qualitative analyses of reasoning behaviors beyond reasoning performance to demonstrate how REPRO has improved the LLM thought.

REPRO Improves the Token Efficiency of Reasoning. We analyze the token cost of REPRO, examining its impact on reasoning efficiency. Figure 3 illustrates the dynamic changes in reasoning token cost during training with and without REPRO. The results show that REPRO effectively reduces token cost as training progresses. Additionally, we compare REPRO’s token cost against baselines in

432 Figure Figure 5, demonstrating significant reductions in inference token cost across all benchmarks.
 433 These findings indicate that REPRO promotes more concise and effective reasoning trajectories,
 434 enhancing both efficiency and practical applicability.

435 **REPRO Reduces Suboptimal Thinking Behaviors.** To gain deeper insight into the impact of
 436 REPRO beyond mere token counts, we analyze changes in the thinking patterns of LLMs. Every 10
 437 steps during the REPRO training process, we instruct the LLM to perform reasoning on the AIME24
 438 benchmark and apply the prompt proposed by Gandhi et al. (2025) (detailed in § C.5) for thinking
 439 pattern recognition. Figure 4 illustrates the evolving proportion of the *backtracking* pattern, which
 440 typically indicates ineffective and excessive reasoning. As training progresses, the prevalence of
 441 this suboptimal pattern significantly decreases compared with vanilla GRPO, highlighting REPRO’s
 442 effectiveness in enhancing the reasoning behaviors and patterns of LLMs.

443 **Case Study.** We also present a qualitative analysis of the responses generated by REPRO, as illustrated
 444 in Case E.1, Case E.2, and Case E.3. Compared to responses from the LLM trained with a vanilla
 445 RL algorithm, REPRO significantly reduces inefficient and suboptimal backtracking and reasoning
 446 (highlighted in orange), resulting in a more linear and efficient thinking process. Furthermore, by
 447 mitigating oscillations around “saddle points”, REPRO-trained models exhibit fewer errors (marked
 448 in red), enhancing overall reasoning accuracy.

450 5 RELATED WORK

451 **Demystifying Reasoning Trajectories of LLMs.** The advent of powerful reasoning LLMs, enhanced
 452 by reasoning trajectories, has spurred extensive research to uncover the underlying mechanisms
 453 within these trajectories. Foundational studies (Yun et al., 2020a;b) have established that sufficiently
 454 expressive Transformers (Vaswani et al., 2017) can act as universal approximators for continuous
 455 sequence-to-sequence mappings over compact domains. Subsequent analyses have explored their
 456 computational power and expressive limitations (Dehghani et al., 2019; Bhattacharya et al., 2020; Yao
 457 et al., 2021; Hewitt et al., 2020; Weiss et al., 2021; Merrill et al., 2022; Chiang et al., 2023; Giannou
 458 et al., 2023; Liu et al., 2023). Recent research has shown that Transformers are capable of meta-
 459 learning optimization algorithms, such as gradient descent, within their forward trajectories (Gatmiry
 460 et al., 2024; Huang et al., 2025b). The most related works (Dai et al., 2023; Liu et al., 2025a) treat
 461 the reasoning trajectories of LLMs as optimization processes for their parameters and internal states,
 462 providing a solid foundation for our work.

463 **Promoting and Improving LLM Reasoning.** RL has emerged as a powerful paradigm for enhancing
 464 the reasoning capabilities of LLMs, with a notable approach being RLVR (OpenAI, 2024a;b; 2025;
 465 DeepSeek-AI et al., 2025; Kimi-Team et al., 2025; Yang et al., 2025; Comanici et al., 2025). Many of
 466 these methods leverage test-time scaling, a process where models engage in iterative self-improvement
 467 by refining their internal thought processes, exploring diverse strategies, and executing self-correction,
 468 often guided by CoT prompting. The resulting models, often termed long-CoT LLMs, have shown
 469 remarkable performance improvements on complex tasks in domains like mathematics, science, and
 470 code. More recent work has focused on refining the RL algorithms themselves. For instance, methods
 471 such as Dr.GRPO (Liu et al., 2025d), VAPO (Yue et al., 2025), and DAPO (Yu et al., 2025a) introduce
 472 algorithmic adaptations, particularly in sampling strategies and advantage estimation, to further
 473 elevate the reasoning performance of LLMs. Other works (Aggarwal & Welleck, 2025; Liu et al.,
 474 2025b; Wu et al., 2025b; Wang et al., 2025a) focuses on introducing new forms of regularization or
 475 rewards based on length or information during training to reduce invalid token consumption.

476 6 CONCLUSION

477 In this paper, we propose REPRO, a novel framework designed to refine the reasoning processes
 478 of LLMs from an optimization perspective. We conceptualize CoT reasoning as an optimization
 479 process and introduce two scores to evaluate its intensity and stability. These scores are integrated
 480 as a process-level reward into the training pipeline of RLVR. Extensive experiments across diverse
 481 reasoning benchmarks demonstrate the effectiveness of REPRO. Furthermore, we illustrate how
 482 REPRO enhances the reasoning behavior of LLMs, improving their efficiency.

486 ETHICS STATEMENT
487488 This study focuses solely on general research tasks and poses no risks to health, safety, personal
489 security, or privacy. No human participants are involved, and no new datasets are released as part
490 of this work. Furthermore, the research does not include potentially harmful insights, methods, or
491 applications, nor does it raise concerns related to privacy, security, legal compliance, or research
492 integrity. Consequently, we anticipate no ethical risks or conflicts of interest. We are committed to
493 maintaining the highest standards of scientific integrity and adhering to ethical guidelines throughout
494 all stages of the research process.
495496 REPRODUCIBILITY STATEMENT
497498 We provide a comprehensive description of the proposed REPRO framework in § 3. To ensure
499 reproducibility, we detail implementation specifics, including datasets, model configurations,
500 and additional information, in §§ C and 4.1. Key code implementations are included in the
501 supplementary materials, with the complete code to be released publicly upon acceptance of the paper.
502503 REFERENCES
504505 Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K.
506 Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett,
507 Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch
508 Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko,
509 Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross,
510 Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu,
511 Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel,
512 Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott
513 Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng
514 Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney,
515 Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex
516 Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo,
517 Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass,
518 Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David
519 Robinson, Bob Rotstед, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit
520 Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet,
521 Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia
522 Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang,
523 Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang,
524 Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card. *CoRR*,
525 abs/2508.10925, 2025. D.3526 Pranjal Aggarwal and Sean Welleck. L1: controlling how long A reasoning model thinks with
527 reinforcement learning. *CoRR*, abs/2503.04697, 2025. 5, A, B, 3528 Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin,
529 Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning
530 from human feedback in llms. In *ACL (1)*, pp. 12248–12267. Association for Computational
531 Linguistics, 2024. 2, C.1532 Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models
533 for mathematical reasoning: Progresses and challenges. In *EACL (Student Research Workshop)*,
534 pp. 225–237. Association for Computational Linguistics, 2024. 1535 Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit
536 Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints.
537 In *EMNLP*, pp. 4895–4901. Association for Computational Linguistics, 2023. D.1

540 Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan,
 541 Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with
 542 large language models. *CoRR*, abs/2108.07732, 2021. 4.1, C.3

543

544 Satwik Bhattacharya, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers
 545 to recognize formal languages. In *EMNLP*, pp. 7096–7116. Association for Computational
 546 Linguistics, 2020. 5, A

547

548 ByteDance Seed. Introduction to techniques used in seed1.6. https://seed/bytedance.com/en/seed1_6, 2025. A

549

550 Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu,
 551 Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-
 552 thought for reasoning large language models. *CoRR*, abs/2503.09567, 2025a. 1

553

554 Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuwei Liu,
 555 Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT
 556 think that much for 2+3=? on the overthinking of o1-like llms. *CoRR*, abs/2412.21187, 2024. 1, A

557

558 Zhipeng Chen, Yingqian Min, Beichen Zhang, Jie Chen, Jinhao Jiang, Daixuan Cheng, Wayne Xin
 559 Zhao, Zheng Liu, Xu Miao, Yang Lu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. An empirical
 560 study on eliciting and improving r1-like reasoning models. *CoRR*, abs/2503.04548, 2025b. C.2

561

562 Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu
 563 Wei. Reasoning with exploration: An entropy perspective. *CoRR*, abs/2506.14758, 2025. 3.4

564

565 David Chiang, Peter Cholak, and Anand Pillay. Tighter bounds on the expressivity of transformer
 566 encoders. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 5544–5562.
 567 PMLR, 2023. 5, A

568

569 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 570 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
 571 Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. D.1

572

573 Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit S.
 574 Dhillon, Marcel Blstein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marrs, Sam Petulla, Colin
 575 Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szekely, Nan-
 576 Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric
 577 Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, Kornraphop Kawintiranon, Tania
 578 Bedrax-Weiss, Oliver Wang, Ya Xu, Ollie Purkiss, Uri Mendlovic, Ilia Deutel, Nam Nguyen,
 579 Adam Langley, Flip Korn, Lucia Rossazza, Alexandre Ramé, Sagar Waghmare, Helen Miller,
 580 Nathan Byrd, Ashrith Sheshan, Raia Hadsell Sangnie Bhardwaj, Pawel Janus, Tero Rissa, Dan
 581 Horgan, Sharon Silver, Ayzaan Wahid, Sergey Brin, Yves Raimond, Klemen Klobcic, Cindy
 582 Wang, Nitesh Bharadwaj Gundavarapu, Ilia Shumailov, Bo Wang, Mantas Pajarskas, Joe Heyward,
 583 Martin Nikoltchev, Maciej Kula, Hao Zhou, Zachary Garrett, Sushant Kafle, Sercan Arik, Ankita
 584 Goel, Mingyao Yang, Jiho Park, Koji Kojima, Parsa Mahmoudieh, Koray Kavukcuoglu, Grace
 585 Chen, Doug Fritz, Anton Bulyenov, Sudeshna Roy, Dimitris Paparas, Hadar Shemtov, Bo-Juen
 586 Chen, Robin Strudel, David Reitter, Aurko Roy, Andrey Vlasov, Changwan Ryu, Chas Leichner,
 587 Haichuan Yang, Zelda Mariet, Denis Vnukov, Tim Sohn, Amy Stuart, Wei Liang, Minmin Chen,
 588 Prayna Rawlani, Christy Koh, JD Co-Reyes, Guangda Lai, Praseem Banzal, Dimitrios Vytiniotis,
 589 Jieru Mei, and Mu Cai. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality,
 590 long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025. 1, 5, A

591

592 Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen
 593 Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen
 594 Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language
 595 models. *CoRR*, abs/2505.22617, 2025. 3.4

596

597 Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT
 598 learn in-context? language models secretly perform gradient descent as meta-optimizers. In *ACL*
 599 (*Findings*), pp. 4005–4019. Association for Computational Linguistics, 2023. 5, A

594 Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated
 595 convolutional networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp.
 596 933–941. PMLR, 2017. [D.1](#)

597

598 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,
 599 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,
 600 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao
 601 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,
 602 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
 603 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,
 604 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang
 605 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong,
 606 Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao,
 607 Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang,
 608 Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang,
 609 Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L.
 610 Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhua Chen, Shengfeng Ye, Shiyu Wang,
 611 Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning
 612 capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. [1](#), [2](#), [3.2](#), [4.1](#), [5](#), [A](#), [B](#)

613 Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal
 614 transformers. In *ICLR*. OpenReview.net, 2019. [5](#), [A](#)

615 Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM learns when to think. *CoRR*,
 616 abs/2505.13379, 2025. [A](#)

617 Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing
 618 the mystery behind chain of thought: A theoretical perspective. In *NeurIPS*, 2023. [1](#)

619

620 Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive
 621 behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *CoRR*,
 622 abs/2503.01307, 2025. [4.4](#), [C.5](#)

623 Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma,
 624 Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan,
 625 Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omnimath:
 626 A universal olympiad level mathematic benchmark for large language models. In *ICLR*.
 627 OpenReview.net, 2025. [C.2](#)

628

629 Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can
 630 looped transformers learn to implement multi-step gradient descent for in-context learning? In
 631 *ICML*. OpenReview.net, 2024. [5](#), [A](#)

632 Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris
 633 Papailiopoulos. Looped transformers as programmable computers. In *ICML*, volume 202 of
 634 *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 2023. [5](#), [A](#)

635

636 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob
 637 Steinhardt. Measuring massive multitask language understanding. In *ICLR*. OpenReview.net,
 638 2021a. [D.1](#)

639

640 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
 641 and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In
 642 *NeurIPS Datasets and Benchmarks*, 2021b. [4.1](#), [C.3](#), [D.1](#)

643

644 John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang, and Christopher D. Manning. Rnns can
 645 generate bounded hierarchical languages with optimal memory. In *EMNLP*, pp. 1978–2010.
 646 Association for Computational Linguistics, 2020. [5](#), [A](#)

647

648 Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang.
 649 Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *CoRR*,
 650 abs/2504.01296, 2025. [A](#), [B](#), [3](#)

648 Jian Hu. REINFORCE++: A simple and efficient approach for aligning large language models.
 649 *CoRR*, abs/2501.03262, 2025. 1, 2, C.1, C.4
 650

651 Jingcheng Hu, Yinmin Zhang, Qi Han, Dixin Jiang, Xiangyu Zhang, and Heung-Yeung Shum.
 652 Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base
 653 model. *CoRR*, abs/2503.24290, 2025. 2

654 Jianhao Huang, Zixuan Wang, and Jason D. Lee. Transformers learn to implement multi-step gradient
 655 descent with chain of thought. In *ICLR*. OpenReview.net, 2025a. 1
 656

657 Jianhao Huang, Zixuan Wang, and Jason D. Lee. Transformers learn to implement multi-step gradient
 658 descent with chain of thought. *CoRR*, abs/2502.21212, 2025b. 5, A

659 Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In
 660 *ACL (Findings)*, pp. 1049–1065. Association for Computational Linguistics, 2023. 1
 661

662 Stuart Hunter. The exponentially weighted moving average. *J. Qual. Technol.*, 18(4):203–210, 1986.
 663 3.3

664 Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando
 665 Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free
 666 evaluation of large language models for code. In *ICLR*. OpenReview.net, 2025. 4.1, C.3, D.1
 667

668 Frederick Jelinek, Robert Mercer, Lalit Bahl, and James Baker. Perplexity—a measure of the difficulty
 669 of speech recognition tasks. *Acoust. Soc. Am.*, 62(S1):S63–S63, 1977. 1

670

671 Zixuan Jiang, Jiaqi Gu, Hanqing Zhu, and David Z. Pan. Pre-rmsnorm and pre-crmsnorm transformers:
 672 Equivalent and efficient pre-ln transformers. In *NeurIPS*, 2023. D.1

673 Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li,
 674 Chengwei Qin, Peifeng Wang, Silvio Savarese, Caiming Xiong, and Shafiq Joty. A survey of
 675 frontiers in LLM reasoning: Inference scaling, learning to reason, and agentic systems. *Trans.
 676 Mach. Learn. Res.*, 2025, 2025. 1

677

678 Maurice Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938. 3.3

679

680 Kimi-Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun
 681 Xiao, Chenzhuang Du, Chonghua Liao, Chunling Tang, Congcong Wang, Dehao Zhang, Enming
 682 Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han
 683 Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li,
 684 Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su,
 685 Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye,
 686 Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu,
 687 Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong,
 688 Weiran He, Weixiao Huang, Wenhai Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingze Wu,
 689 Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang
 690 Liu, Yanru Chen, Yeqie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du,
 691 Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu
 692 Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling
 693 reinforcement learning with llms. *CoRR*, abs/2501.12599, 2025. 1, 5, A

694

695 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph
 696 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
 697 serving with pagedattention. In *SOSP*, pp. 611–626. ACM, 2023. B, C.4

698

699 Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A
 700 simple, effective, and efficient reinforcement learning method for aligning large language models.
 701 In *ICML*. OpenReview.net, 2024. 2, C.1

702

703 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan
 704 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *ICLR*.
 705 OpenReview.net, 2024. 1, B, C.3

702 Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers
 703 learn shortcuts to automata. In *ICLR*. OpenReview.net, 2023. 5, A
 704

705 Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang,
 706 Songyang Zhang, and Kai Chen. Are your llms capable of stable reasoning? *CoRR*, abs/2412.13147,
 707 2024. 4.1, A, C.3

708 Junnan Liu, Hongwei Liu, Linchen Xiao, Shudong Liu, Taolin Zhang, Zihan Ma, Songyang Zhang,
 709 and Kai Chen. Deciphering trajectory-aided LLM reasoning: An optimization perspective. *CoRR*,
 710 abs/2505.19815, 2025a. 1, 3.1, 5, A
 711

712 Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang,
 713 and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. *CoRR*,
 714 abs/2505.15612, 2025b. 5, A, B, 3

715 Zechun Liu, Ernie Chang, Changsheng Zhao, Chia-Jung Chang, Wei Wen, Chen Lai, Rick Cao,
 716 Yuandong Tian, Raghuraman Krishnamoorthi, Yangyang Shi, and Vikas Chandra. Mobilellm-r1:
 717 Model card. <https://huggingface.co/mobilellm-r1>, 2025c. 4.1, D.1
 718

719 Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min
 720 Lin. Understanding r1-zero-like training: A critical perspective. *CoRR*, abs/2503.20783, 2025d. 5,
 721 A, C.1

722 Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta,
 723 Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler:
 724 Surpassing o1-preview with a 1.5b model by scaling rl. <https://pretty-radio-b75.notion.site/DeepScalE-R-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2>, 2025. Notion
 725 Blog. 4.1, C.2

726 William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth
 727 threshold circuits. *Trans. Assoc. Comput. Linguistics*, 10:843–856, 2022. 5, A

728 Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>, 2025. Accessed: 2025-
 729 04. D.1

730 OpenAI. Learning to reason with llms. <https://openai.com/index/learning-to-reason-with-llms/>, 2024a. Accessed: 2024-09. 1, 5, A

731 OpenAI. Introducing openai o3 and o4-mini. <https://openai.com/index/introducing-o3-and-o4-mini/>, 2024b. Accessed: 2024-12. 1, 5, A

732 OpenAI. Gpt-5 and the new era of work. <https://openai.com/index/gpt-5-new-era-of-work/>, 2025. Accessed: 2025-08. 1, 5, A

733 Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong
 734 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton,
 735 Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and
 736 Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*,
 737 2022. 2

738 Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck.
 739 Reasoning with large language models, a survey. *CoRR*, abs/2407.11511, 2024. 1

740 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani,
 741 Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark.
 742 *CoRR*, abs/2311.12022, 2023. 4.1, C.3

743 John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-
 744 dimensional continuous control using generalized advantage estimation. In *ICLR (Poster)*, 2016.
 745 2

756 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
 757 optimization algorithms. *CoRR*, abs/1707.06347, 2017. 1, 2, C.4
 758

759 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li,
 760 Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open
 761 language models. *CoRR*, abs/2402.03300, 2024. 1, 2, C.1, C.4
 762

763 Noam Shazeer. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150,
 764 2019. B
 765

766 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,
 767 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *EuroSys*,
 768 pp. 1279–1297. ACM, 2025. C.4
 769

770 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally
 771 can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. 1
 772

773 Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer:
 774 Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. D.1
 775

776 Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu,
 777 Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Ben Hu. Stop overthinking: A survey on
 778 efficient reasoning for large language models. *CoRR*, abs/2503.16419, 2025. 1
 779

780 Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu,
 781 Mingyu Ding, Hongyang Li, Mengzhe Geng, Yue Wu, Wenhui Wang, Junsong Chen, Zhangyue
 782 Yin, Xiaozhe Ren, Jie Fu, Junxian He, Wu Yuan, Qi Liu, Xihui Liu, Yu Li, Hao Dong, Yu Cheng,
 783 Ming Zhang, Pheng-Ann Heng, Jifeng Dai, Ping Luo, Jingdong Wang, Ji-Rong Wen, Xipeng Qiu,
 784 Yike Guo, Hui Xiong, Qun Liu, and Zhenguo Li. A survey of reasoning with foundation models:
 785 Concepts, methodologies, and outlook. *ACM Comput. Surv.*, 57(11):278:1–278:43, 2025. 1
 786

787 Yunhao Tang, Sid Wang, and Rémi Munos. Learning to chain-of-thought with jensen’s evidence
 788 lower bound. *CoRR*, abs/2503.19618, 2025. 3.2
 789

790 Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL
 791 <https://qwenlm.github.io/blog/qwq-32b/>. A
 792

793 Tencent. Hunyuan-1.8b. <https://github.com/Tencent-Hunyuan/Hunyuan-1.8B/>,
 794 2025. 4.1, D.1
 795

796 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz
 797 Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017. 5, A
 798

799 Jingyao Wang, Wenwen Qiang, Zeen Song, Changwen Zheng, and Hui Xiong. Learning to think:
 800 Information-theoretic reinforcement fine-tuning for llms. *CoRR*, abs/2505.10425, 2025a. 3.1, 5, A
 801

802 Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang
 803 Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*
 804 (1), pp. 9426–9439. Association for Computational Linguistics, 2024. B
 805

806 Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen,
 807 Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen
 808 Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive
 809 effective reinforcement learning for LLM reasoning. *CoRR*, abs/2506.01939, 2025b. 3.4
 810

811 Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu,
 812 Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Thoughts are all
 813 over the place: On the underthinking of o1-like llms. *CoRR*, abs/2501.18585, 2025c. 1, A
 814

815 Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In *ICML*, volume 139 of
 816 *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 2021. 5, A
 817

818 Siye Wu, Jian Xie, Yikai Zhang, Aili Chen, Kai Zhang, Yu Su, and Yanghua Xiao. ARM: adaptive
 819 reasoning model. *CoRR*, abs/2505.20258, 2025a. A

810 Xingyu Wu, Yuchen Yan, Shangke Lyu, Linjuan Wu, Yiwen Qiu, Yongliang Shen, Weiming Lu, Jian
 811 Shao, Jun Xiao, and Yueting Zhuang. LAPO: internalizing reasoning efficiency via length-adaptive
 812 policy optimization. *CoRR*, abs/2507.15758, 2025b. 5, A

813 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang
 814 Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu,
 815 Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang,
 816 Jiaxi Yang, Jingren Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu,
 817 Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men,
 818 Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren,
 819 Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Ying Zhang,
 820 Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu.
 821 Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. 4.1, 5, A, C.5, D.1, D.2, D.3

822 Shunyu Yao, Binghui Peng, Christos H. Papadimitriou, and Karthik Narasimhan. Self-attention net-
 823 works can process bounded hierarchical languages. In *ACL/IJCNLP*, pp. 3770–3785. Association
 824 for Computational Linguistics, 2021. 5, A

825 Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong
 826 Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi
 827 Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi
 828 Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying
 829 Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source
 830 LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025a. 5, A, C.1

831 Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao,
 832 Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. RLPR: extrapolating RLVR to general domains
 833 without verifiers. *CoRR*, abs/2506.18254, 2025b. 3.2

834 Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Cheng-Xiang
 835 Wang, Tiantian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun
 836 Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang,
 837 Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. VAPO: efficient and reliable reinforcement
 838 learning for advanced reasoning tasks. *CoRR*, abs/2504.05118, 2025. 5, A, C.1

839 Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are
 840 transformers universal approximators of sequence-to-sequence functions? In *ICLR*. OpenRe-
 841 view.net, 2020a. 5, A

842 Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and
 843 Sanjiv Kumar. O(n) connections are expressive enough: Universal approximability of sparse
 844 transformers. In *NeurIPS*, 2020b. 5, A

845 Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can
 846 learn when to think. *CoRR*, abs/2505.13417, 2025. A, 3

847 Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong
 848 Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization.
 849 *CoRR*, abs/2507.18071, 2025. C.1

850 Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao,
 851 Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. Sqlang:
 852 Efficient execution of structured language model programs. In *NeurIPS*, 2024. B

853 Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu,
 854 Yanjun Lyu, Peng Shu, Xiaowei Yu, Chao Cao, Hanqi Jiang, Hanxu Chen, Yiwei Li, Junhao Chen,
 855 Huawei Hu, Yihen Liu, Huaqin Zhao, Shaochen Xu, Haixing Dai, Lin Zhao, Ruidong Zhang, Wei
 856 Zhao, Zhenyuan Yang, Jingyuan Chen, Peilong Wang, Wei Ruan, Hui Wang, Huan Zhao, Jing
 857 Zhang, Yiming Ren, Shihuan Qin, Tong Chen, Jiaxi Li, Arif Hassan Zidan, Afrar Jahin, Minheng
 858 Chen, Sichen Xia, Jason Holmes, Yan Zhuang, Jiaqi Wang, Bochen Xu, Weiran Xia, Jichao Yu,
 859 Kaibo Tang, Yaxuan Yang, Bolun Sun, Tao Yang, Guoyu Lu, Xianqiao Wang, Lilong Chai, He Li,
 860 Jin Lu, Lichao Sun, Xin Zhang, Bao Ge, Xintao Hu, Lian Zhang, Hua Zhou, Lu Zhang, Shu Zhang,

864 Ninghao Liu, Bei Jiang, Linglong Kong, Zhen Xiang, Yudan Ren, Jun Liu, Xi Jiang, Yu Bao, Wei
865 Zhang, Xiang Li, Gang Li, Wei Liu, Dinggang Shen, Andrea Sikora, Xiaoming Zhai, Dajiang
866 Zhu, and Tianming Liu. Evaluation of openai o1: Opportunities and challenges of AGI. *CoRR*,
867 abs/2409.18486, 2024. [1](#)

868 Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang,
869 Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. *CoRR*, abs/2505.21493,
870 2025. [3.2](#)

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

Appendix

Table of Contents

A More Detailed Related Work	19
B Discussions	19
C More Implementation Details	20
C.1 Implementation of Normalization Rectifying Process Reward in Critic-Free RL	20
C.2 Training Data	21
C.3 Evaluation Benchmarks	21
C.4 Training Parameters	21
C.5 Prompt for Thinking Pattern Recognition	23
D Additional Experimental Results	23
D.1 REPRO on LLMs of Diverse Families	23
D.2 REPRO on Larger LLMs	25
D.3 REPRO for zero-RLVR	25
D.4 Ablation of the Number of Selected Segments	26
D.5 Entropy Dynamics of REPRO	27
E Cases	27
F Limitations	41
G LLM Usage	41

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972 A MORE DETAILED RELATED WORK
973

974 **Demystifying Reasoning Trajectories of LLMs.** The advent of powerful reasoning LLMs, enhanced
975 by reasoning trajectories, has spurred extensive research to uncover the underlying mechanisms
976 within these trajectories. Foundational studies (Yun et al., 2020a;b) have established that sufficiently
977 expressive Transformers (Vaswani et al., 2017) can act as universal approximators for continuous
978 sequence-to-sequence mappings over compact domains. Subsequent analyses have explored their
979 computational power and expressive limitations (Dehghani et al., 2019; Bhattacharya et al., 2020; Yao
980 et al., 2021; Hewitt et al., 2020; Weiss et al., 2021; Merrill et al., 2022; Chiang et al., 2023; Giannou
981 et al., 2023; Liu et al., 2023). Recent research has shown that Transformers are capable of meta-
982 learning optimization algorithms, such as gradient descent, within their forward trajectories (Gatmiry
983 et al., 2024; Huang et al., 2025b). The most related works (Dai et al., 2023; Liu et al., 2025a) treat
984 the reasoning trajectories of LLMs as optimization processes for their parameters and internal states,
985 providing a solid foundation for our work.

986 **Promoting and Improving LLM Reasoning.** RL has emerged as a powerful paradigm for enhancing
987 the reasoning capabilities of LLMs, with a notable approach being RLVR (OpenAI, 2024a;b; 2025;
988 DeepSeek-AI et al., 2025; Kimi-Team et al., 2025; Team, 2025; Yang et al., 2025; Comanici et al.,
989 2025; ByteDance Seed, 2025). Many of these methods leverage test-time scaling, a process where
990 models engage in iterative self-improvement by refining their internal thought processes, exploring
991 diverse strategies, and executing self-correction, often guided by CoT prompting. The resulting
992 models, often termed long-CoT LLMs, have shown remarkable performance improvements on
993 complex tasks in domains like mathematics, science, and code. More recent work has focused on
994 refining the RL algorithms themselves. For instance, methods such as Dr.GRPO (Liu et al., 2025d),
995 VAPO (Yue et al., 2025), and DAPO (Yu et al., 2025a) introduce algorithmic adaptations, particularly
996 in sampling strategies and advantage estimation, to further elevate the reasoning performance of
997 LLMs. A significant limitation of long-CoT LLMs is their computational inefficiency, often resulting
998 in “overthinking”, characterized by the generation of redundant tokens or unnecessary reasoning
999 steps that may lead to errors (Chen et al., 2024; Liu et al., 2024; Wang et al., 2025c). To address
1000 this issue, one research direction (Hou et al., 2025; Aggarwal & Welleck, 2025; Liu et al., 2025b;
1001 Wu et al., 2025b; Wang et al., 2025a) focuses on introducing new forms of regularization or rewards
1002 based on length or information during training to reduce invalid token consumption. Another research
1003 approach (Yang et al., 2025; Fang et al., 2025; Wu et al., 2025a; Zhang et al., 2025) aims to learn
1004 adaptive policies that control the reasoning process by altering the reasoning pattern according to
1005 question difficulty or user instructions.

1006 B DISCUSSIONS
1007

1008 **Computational Efficiency.** While REPRO necessitates additional computation due to the entropy-
1009 based selection strategy and reward calculation, these forward processes exhibit significant prefix
1010 overlap. Modern LLM inference engines like vLLM (Kwon et al., 2023) and SGLang (Zheng et al.,
1011 2024) expedite forward computation by caching key-value pairs (KV cache) (Shazeer, 2019). In our
1012 training process, we utilize the inference engines supported by the training framework to accelerate
1013 the computation of \mathcal{S} .

1014 **Comparison with Process Reward Models.** Process reward models (Lightman et al., 2024; Wang
1015 et al., 2024) have been proposed to offer process-level granularity in supervising the training of
1016 reasoning LLMs. However, these methods struggle to generate effective supervision signals due
1017 to the complex dependencies among different processes and the final answers (DeepSeek-AI et al.,
1018 2025). In contrast, the method proposed in this paper, REPRO, neither relies on additional models
1019 nor provides absolute supervision signals of correctness or incorrectness. Instead, it evaluates the
1020 contribution of each process to reasoning and offers relative advantages and disadvantages through
1021 group normalization, significantly reducing noise in the signals provided for training.

1022 **Comparison with Efficient Reasoning Methods.** Recent studies on efficient LLM reasoning
1023 (Aggarwal & Welleck, 2025; Hou et al., 2025; Liu et al., 2025b) emphasize the inclusion of a penalty
1024 coefficient for response length during the post-training process of LLMs, which rewards shorter,
1025 correct trajectories and penalizes longer ones. These methods only perform a coarse-grained length-

1026 based evaluation of the model’s trajectories, which can easily lead to incorrect penalties for necessary
 1027 and correct trajectories, thereby having a negative impact on the model’s performance (Liu et al.,
 1028 2025b). Alternatively, our method provides a more nuanced process reward, enhancing the LLMs’
 1029 capability and efficiency by refining its reasoning patterns with finer granularity. As shown in Table 3,
 1030 REPRO outperforms salient baselines focusing on reasoning performance and efficiency. Compared
 1031 to the Vanilla GRPO baseline, REPRO achieves superior performance across all benchmarks (e.g.,
 1032 +3.1% on AIME24) while reducing token consumption by approximately 50% (12,367 to 6,158
 1033 tokens). In contrast to efficiency-focused methods like ThinkPrune and Laser, which suffer significant
 1034 performance drops to achieve lower latency, REPRO successfully decouples efficiency from accuracy
 1035 degradation. For instance, while L1-Max matches our token efficiency, it severely lags in reasoning
 1036 capability (26.7% vs. 36.0% on AIME24), highlighting REPRO’s ability to condense reasoning
 1037 without information loss. Similarly, REPRO surpasses AdaThink by a clear margin of 4.3% on
 1038 AIME24 while requiring 28% fewer tokens, demonstrating a more effective internal optimization
 1039 than external stopping criteria.

1040 Table 3: Comparison between REPRO and ThinkPrune (Hou et al., 2025), L1 (Aggarwal & Welleck,
 1041 2025), Laser (Liu et al., 2025b), and AdaThink (Zhang et al., 2025).

Methods	AIME24	AIME25	MATH500	GPQA-D	Avg #Tokens
	% Avg@16 ↑	% Avg@16 ↑	% Avg@4 ↑	% Avg@4 ↑	
Original	30.6	24.8	84.4	32.7	10,089
Vanilla GRPO	32.9	25.3	86.0	34.5	12,367
ThinkPrune-4k	31.7	20.0	84.5	33.2	8,376
L1-Max	26.7	17.9	85.3	34.1	6,249
Laser-D-L4096	27.1	17.5	85.0	34.2	5,713
Laser-DE-L4096	27.1	22.5	84.7	32.5	5,862
AdaThink _{δ0.05}	31.7	25.9	82.5	34.1	8,549
REPRO	36.0	26.5	87.1	37.0	6,558

C MORE IMPLEMENTATION DETAILS

C.1 IMPLEMENTATION OF NORMALIZATION RECTIFYING PROCESS REWARD IN CRITIC-FREE RL

In this section, we present the implementation details for normalizing rectifying process rewards in several representative critic-free RL algorithms.

GRPO and Its Variants. For GRPO (Shao et al., 2024), normalization is applied across all segments within the trajectory group G for each question q , as follows:

$$\tilde{r}'_{j,i} = \frac{\tilde{r}_{j,i} - \text{mean}(\{\tilde{r}_{l,m}\}_{1 \leq l \leq k, 1 \leq m \leq [G]})}{\text{std}(\{\tilde{r}_{l,m}\}_{1 \leq l \leq k, 1 \leq m \leq [G]})}, \quad (15)$$

where $\tilde{r}'_{j,i}$ denotes the normalized reward for the j -th segment of the i -th trajectory in the group G . This normalization also applies to the variants of GRPO, such as Dr.GRPO (Liu et al., 2025d), VAPO (Yue et al., 2025), DAPO (Yu et al., 2025a), and GSPO (Zheng et al., 2025).

REINFORCE++. In contrast to GRPO, REINFORCE++ (Hu, 2025) normalizes the reward for each segment j of each trajectory i across the full batch \mathcal{B} :

$$\tilde{r}'_{j,i} = \frac{\tilde{r}_{j,i} - \text{mean}(\{\tilde{r}_{l,m}\}_{1 \leq l \leq k, 1 \leq m \leq [\mathcal{B}]})}{\text{std}(\{\tilde{r}_{l,m}\}_{1 \leq l \leq k, 1 \leq m \leq [\mathcal{B}]})}, \quad (16)$$

considering all segments in all trajectories within the batch \mathcal{B} .

RLOO. Similar to GRPO, RLOO (Ahmadian et al., 2024) performs normalization within the group G for each question q as follows:

$$\tilde{r}'_{j,i} = \tilde{r}_{j,i} - \frac{1}{k(G-1)} \sum_{l \in [k]} \sum_{m \neq i, m \in [G]} \tilde{r}_{l,m}. \quad (17)$$

1080 **ReMax.** In ReMax (Li et al., 2024), we normalize the rectifying process reward by the mean of all
 1081 rectifying process rewards from the trajectory generated by greedy decoding:
 1082

$$1083 \tilde{r}'_{j,i} = \tilde{r}_{j,i} - \frac{1}{k} \sum_{l \in [k]} \tilde{r}_l, \quad (18)$$

1085 where \tilde{r}_l denotes the rectifying process rewards of the trajectory generated by greedy decoding.
 1086

1087 C.2 TRAINING DATA

1089 We utilize DeepScaleR-Preview-Dataset proposed in Luo et al. (2025) for all model training. The
 1090 dataset consists of approximately 40,000 unique mathematics problem-answer pairs compiled from:
 1091

- 1092 • American Invitational Mathematics Examination problems (1984-2023).
- 1093 • American Mathematics Competition problems (before 2023).
- 1094 • Omni-MATH dataset (Gao et al., 2025).
- 1095 • Still dataset (Chen et al., 2025b).

1098 C.3 EVALUATION BENCHMARKS

1100 The following details describe our evaluation benchmarks:

- 1101 • **AIME24.** This dataset consists of 30 challenging problems from the 2024 American Invitational
 Mathematics Examination.
- 1104 • **AIME25.** This dataset consists of 30 challenging problems from the 2025 American Invitational
 Mathematics Examination.
- 1106 • **MATH500.** The original MATH dataset (Hendrycks et al., 2021b) contains 12,500 problems
 from American high school mathematics competitions. MATH500 (Lightman et al., 2024), a
 widely used subset of its test split, includes only Level 5 questions, which we adopt in this paper.
- 1109 • **LiveMathBench.** LiveMathBench (Liu et al., 2024) is a continuously updated benchmark of
 challenging mathematical problems. We use the 202505 hard split, which contains 100 high-
 quality English questions.
- 1112 • **GPQA.** The Graduate-Level Google-Proof Q&A Benchmark (GPQA) (Rein et al., 2023) is a
 multiple-choice science question-answering dataset designed to be resistant to web search. We
 evaluate on its *diamond* subset, which comprises 198 questions.
- 1115 • **MBPP.** The Mostly Basic Programming Problems (MBPP) dataset (Austin et al., 2021) evaluates
 programming models on elementary Python tasks. It was created via crowdsourcing, with workers
 generating problems and solutions under specified guidelines. Problem statements were later
 refined to remove ambiguity, and selected items underwent manual review and editing to ensure
 clarity and accuracy of test cases.
- 1119 • **LiveCodeBench.** LiveCodeBench (Jain et al., 2025) is designed to provide a comprehensive and
 contamination-free evaluation of the coding abilities of large language models. It incorporates
 problems from LeetCode, AtCoder, and Codeforces.

1124 C.4 TRAINING PARAMETERS

1125 We set the hyperparameters $w = 0.5$ and $\alpha = 0.1$, selecting the top-10 segments for each reasoning
 1126 trajectory. Training utilized the veRL (Sheng et al., 2025) and vLLM (Kwon et al., 2023) frameworks.
 1127 Table 4, Table 5, and Table 6 present the training parameters for PPO (Schulman et al., 2017),
 1128 GRPO (Shao et al., 2024), and REINFORCE++ baselines (Hu, 2025), respectively.
 1129

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

Table 4: Training Parameters of PPO.

Parameters	Values
Batch Size	256
Number of Rollout Per Question	8
Rollout Temperature	1.0
Rollout Top- p	1.0
Maximum Number of Generation Tokens	16384
Learning Rate	1e-6
KL Loss Coefficient	0.001
ϵ_{\min}	0.2
ϵ_{\max}	0.28
λ	1.0
γ	1.0
Gradient Clipping	1.0
Number of Training Steps	500

Table 5: Training Parameters of GRPO and REINFORCE++ Baseline.

Parameters	Values
Batch Size	256
Number of Rollout Per Question	8
Rollout Temperature	1.0
Rollout Top- p	1.0
Maximum Number of Generation Tokens	16384
Learning Rate	1e-6
KL Loss Coefficient	0.001
ϵ_{\min}	0.2
ϵ_{\max}	0.28
Gradient Clipping	1.0
Number of Training Steps	500

Table 6: Training Parameters of REINFORCE++.

Parameters	Values
Batch Size	256
Number of Rollout Per Question	1
Rollout Temperature	1.0
Rollout Top- p	1.0
Maximum Number of Generation Tokens	16384
Learning Rate	1e-6
KL Loss Coefficient	0.001
ϵ_{\min}	0.2
ϵ_{\max}	0.28
Gradient Clipping	1.0
Number of Training Steps	500

1188
1189

C.5 PROMPT FOR THINKING PATTERN RECOGNITION

1190 Prompt C.1 illustrates the prompt proposed in [Gandhi et al. \(2025\)](#) for recognizing beneficial thinking
1191 patterns in the reasoning process. In this paper, we utilize Qwen3-235B-A22B-Instruct-2507 ([Yang](#)
1192 et al., 2025) to perform the recognition

1193

1194

Prompt C.1: Prompt for Thinking Pattern Recognition

1195

1196

1197

1198

Below is a chain-of-reasoning generated by a Language Model when attempting to solve a math problem. Evaluate this chain-of-reasoning to determine whether it demonstrates beneficial problem-solving behaviors that deviate from typical linear, monotonic reasoning patterns commonly observed in language models.

1199

1200

1201

1202

```
<start_of_reasoning>
```

```
{input}
```

```
<end_of_reasoning>
```

Specifically, actively identify and emphasize beneficial behaviors such as:

1203

1204

1205

1206

1207

1208

1209

1210

1211

- **Backtracking:** Explicitly revising approaches upon identifying errors or dead ends (e.g., "This approach won't work because...").
- **Verification:** Systematically checking intermediate results or reasoning steps (e.g., "Let's verify this result by...").
- **Subgoal Setting:** Breaking down complex problems into smaller, manageable steps (e.g., "To solve this, we first need to...").
- **Enumeration:** Solving problems by exhaustively considering multiple cases or possibilities.

1212

1213

1214

Additionally, remain attentive to and encourage the identification of other beneficial behaviors not explicitly listed here, such as creative analogies, abstraction to simpler cases, or insightful generalizations.

1215

1216

1217

Important: Clearly specify each beneficial behavior you identify. Provide explicit examples from the reasoning chain. If no beneficial behaviors are observed, explicitly return an empty list.

1218

Provide your evaluation clearly, formatted as follows:

1219

1220

D ADDITIONAL EXPERIMENTAL RESULTS

1221

1222

1223

D.1 REPRO ON LLMS OF DIVERSE FAMILIES

1224

1225

1226

In this section, to further verify the universality and generalization ability of REPRO, we conduct experiments on LLMs of different families.

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

LLMs. We include three LLMs in our experiments: Qwen3-1.7B ([Yang et al., 2025](#)), Hunyuan-1.8B-Instruct ([Tencent, 2025](#)), and MobileLLM-R1-950M ([Liu et al., 2025c](#)). Qwen3-1.7B, part of the Qwen3 series, is a transformer-based dense LLM with 28 layers and a 32k context length, incorporating Grouped Query Attention ([Ainslie et al., 2023](#)), SwiGLU ([Dauphin et al., 2017](#)), Rotary Positional Embeddings ([Su et al., 2024](#)), and RMSNorm ([Jiang et al., 2023](#)). Hunyuan-1.8B-Instruct, from the Hunyuan series, is a 32-layer transformer-based dense LLM that also employs Grouped Query Attention ([Ainslie et al., 2023](#)) and supports a 256K context window, maintaining stable performance on long-text tasks. MobileLLM-R1-950M, from the MobileLLM series, is an efficient reasoning model based on the Llama4 ([Meta, 2025](#)) architecture. Pre-trained on approximately 2T high-quality tokens and with fewer than 5T total training tokens, MobileLLM-R1-950M achieves performance comparable or superior to Qwen3-0.6B, which was trained on 36T tokens, across benchmarks such as MATH ([Hendrycks et al., 2021b](#)), GSM8K ([Cobbe et al., 2021](#)), MMLU ([Hendrycks et al., 2021a](#)), and LiveCodeBench ([Jain et al., 2025](#)). We utilize the same hyperparameters as shown in Table 5.

1240

1241

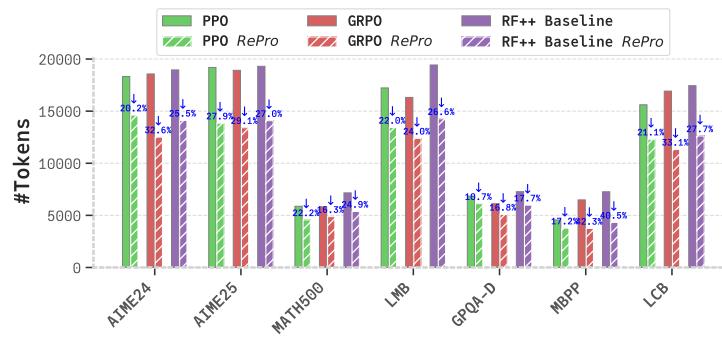
Performance. As shown in Table 7, REPRO consistently enhances performance across various LLM families. For Qwen3-1.7B, integrating REPRO with reinforcement learning algorithms such as PPO, REINFORCE++ Baseline (RF++B), and GRPO yields substantial improvements across nearly

1242 Table 7: Performance of REPRO on LLMs of diverse families. ♠ denotes the in-domain evaluation
 1243 benchmark and ♣ denotes the out-of-domain benchmark.

1245 Methods	1246 AIME24 ♠ % Avg@16 ↑	1246 AIME25 ♠ % Avg@16 ↑	1246 MATH500 ♠ % Avg@4 ↑	1246 LMB ♠ % Avg@4 ↑	1246 GPQA-D ♣ % Avg@4 ↑	1246 MBPP ♣ % Avg@4 ↑	1246 LCB ♣ % Avg@4 ↑
<i>Qwen3-1.7B</i>							
Original	46.8	36.1	91.5	18.8	39.5	66.9	30.6
PPO	45.2	36.5	92.1	20.5	40.2	68.2	31.3
+REPRO	49.0	37.5	92.4	18.8	40.3	69.1	31.5
RF++ B	46.5	34.4	91.5	18.5	38.5	66.1	31.1
+REPRO	49.8	39.0	92.7	22.0	39.8	71.7	32.4
GRPO	47.3	34.8	93.4	18.8	38.3	67.5	30.2
+REPRO	49.8	37.9	94.1	19.5	39.1	68.8	32.0
<i>Hunyuan-1.8B-Instruct</i>							
Original	38.8	32.8	81.3	15.0	38.0	73.1	24.5
PPO	42.1	33.3	86.0	19.5	42.1	75.8	25.9
+REPRO	43.5	32.1	84.3	20.5	42.7	76.2	26.3
RF++ B	42.5	33.3	85.5	17.5	42.4	76.3	26.5
+REPRO	44.3	32.8	86.2	17.8	43.1	77.0	26.2
GRPO	43.3	32.7	85.6	16.0	43.6	75.9	27.1
+REPRO	44.6	33.5	84.2	17.5	44.8	77.5	27.7
<i>MobileLLM-R1-950M</i>							
Original	15.8	18.1	76.4	10.5	19.2	58.2	18.0
PPO	23.2	22.5	81.4	14.5	22.6	63.6	19.8
+REPRO	24.2	23.1	83.5	15.8	24.1	65.1	21.4

1266 all benchmarks. Specifically, PPO+REPRO improves AIME24 performance from 45.2 to 49.0 and
 1267 AIME25 from 36.5 to 37.5, while RF++B+REPRO further boosts AIME25 to 39.0, demonstrating
 1268 its effectiveness on challenging mathematical reasoning tasks. Comparable gains, up to +1.5 points,
 1269 are observed on MBPP and LiveCodeBench. For Hunyuan-1.8B-Instruct, REPRO also delivers
 1270 improvements; for instance, GRPO+REPRO enhances AIME24 from 43.3 to 44.6 and AIME25 from
 1271 32.7 to 33.5, while RF++B+REPRO increases AIME24 from 42.5 to 44.3 and MBPP from 76.3
 1272 to 77.0. Similarly, for the efficiency-oriented MobileLLM-R1-950M, REPRO provides consistent
 1273 benefits, with PPO+REPRO improving AIME24 from 23.2 to 24.2 and AIME25 from 22.5 to 23.1.
 1274 On broader reasoning tasks like MATH500 and GPQA-Diamond, REPRO achieves gains ranging
 1275 from +1.0 to +1.6 points. These results confirm that REPRO is effective not only for larger-scale
 1276 dense models but also generalizes to compact, efficiency-optimized architectures.

1277 **Token Efficiency.** We evaluate the inference token efficiency of REPRO across LLMs from different
 1278 families. As shown in Figures 7 and 8, REPRO consistently outperforms baseline methods in terms
 1279 of token efficiency. For instance, PPO+REPRO requires fewer tokens than standalone PPO while
 1280 achieving higher performance, demonstrating REPRO’s ability to enhance reasoning efficiency across
 1281 diverse LLM architectures.



1295 Figure 7: Comparison of the inference reasoning token cost of REPRO and baselines on Qwen3-1.7B.

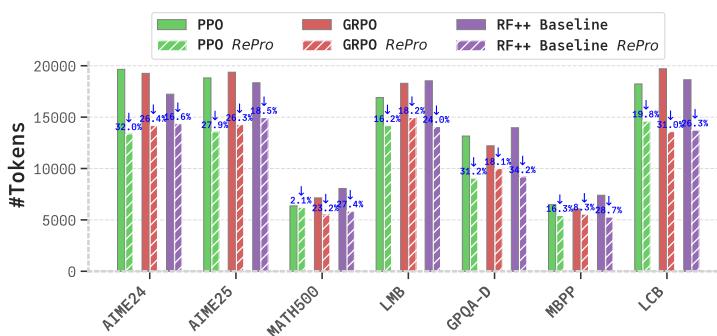


Figure 8: Comparison of the inference reasoning token cost of REPRO and baselines on Hunyuan-1.8B-Instruct.

D.2 REPRO ON LARGER LLMs

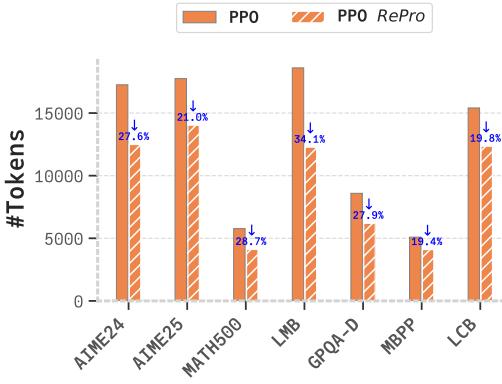


Figure 9: Comparison of the inference reasoning token cost of REPRO and baselines on Qwen3-8B.

In this section, to further verify the effectiveness of REPRO on LLMs of larger scale, we conduct experiments on LLMs of larger scale.

LLMs. We include Qwen3-8B (Yang et al., 2025) in our experiments. Qwen3-8B, part of the Qwen3 series, is a dense LLM with 36 layers and a 128k context length. We utilize the hyperparameters specified in Table 5 for training.

Performance. As shown in Table 8, REPRO consistently outperforms baseline methods when applied to Qwen3-8B. For example, GRPO+REPRO enhances performance on AIME24 from 75.6 to 76.1 and on AIME25 from 67.9 to 68.5, demonstrating improved mathematical reasoning capabilities. Significant gains are also observed in science and code reasoning tasks: GPQA-Diamond improves from 59.5 to 60.4, MBPP from 68.8 to 72.1, and LiveCodeBench from 52.2 to 53.4. These results confirm that REPRO delivers consistent performance improvements across mathematical, scientific, and coding reasoning tasks, even when applied to larger-scale models like Qwen3-8B.

Token Efficiency. As shown in Figure 9, REPRO consistently outperforms vanilla GRPO in terms of token efficiency when applied to Qwen3-8B. For instance, GRPO+REPRO requires fewer tokens than standalone GRPO while achieving higher performance, demonstrating REPRO’s ability to enhance reasoning efficiency across larger-scale models.

D.3 REPRO FOR ZERO-RLVR

Previous discussions and experiments related to REPRO were mostly based on LLMs with deep thinking capabilities. In this section, we also conduct relevant experiments on zero-RLVR trained starting from base LLMs.

1350 Table 8: Performance of REPRO on evaluation reasoning benchmarks with Qwen3-8B. ♠ denotes the
 1351 in-domain evaluation benchmark and ♣ denotes the out-of-domain benchmark.

Methods	AIME24 ♠ % Avg@16 ↑	AIME25 ♠ % Avg@16 ↑	MATH500 ♠ % Avg@4 ↑	LMB ♠ % Avg@4 ↑	GPQA-D ♣ % Avg@4 ↑	MBPP ♣ % Avg@4 ↑	LCB ♣ % Avg@4 ↑
<i>Qwen3-8B</i>							
Original	75.2	66.5	96.8	35.3	58.7	70.0	49.8
GRPO	75.6	67.9	97.5	35.1	59.5	68.8	52.2
+REPRO	76.1	68.5	97.2	35.8	60.4	72.1	53.4

1359 Table 9: Performance of REPRO on evaluation reasoning benchmarks with Qwen3-4B-Base. ♠
 1360 denotes the in-domain evaluation benchmark and ♣ denotes the out-of-domain benchmark.

Methods	AIME24 ♠ % Avg@16 ↑	AIME25 ♠ % Avg@16 ↑	MATH500 ♠ % Avg@4 ↑	LMB ♠ % Avg@4 ↑	GPQA-D ♣ % Avg@4 ↑	MBPP ♣ % Avg@4 ↑	LCB ♣ % Avg@4 ↑
<i>Qwen3-4B-Base</i>							
Original	11.5	7.9	68.3	7.5	16.7	25.0	15.9
GRPO	23.5	19.1	83.6	12.5	39.7	58.7	15.3
+REPRO	21.0	16.7	83.0	14.5	40.5	59.3	17.3

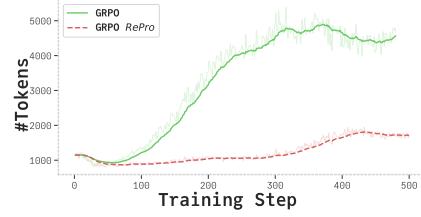
1369 **LLMs.** To be more specific, we train Qwen3-4B-Base model with GRPO and GRPO + REPRO, we
 1370 utilize the same hyperparameters as shown in Table 5.

1372 **Performance.** As presented in Table 9, REPRO and
 1373 GRPO achieve comparable performance on both in-
 1374 domain and out-of-domain benchmarks. In AIME24,
 1375 GRPO slightly outperforms REPRO, but REPRO surpasses
 1376 GRPO in AIME25. Both methods perform comparably in
 1377 MATH500, with REPRO showing a marginal advantage.
 1378 REPRO demonstrates stronger performance in LiveMath-
 1379 Bench and GPQA-Diamond, significantly outperforming
 1380 the Original method. In MBPP and LiveCodeBench, RE-
 1381 PRO and GRPO perform closely, with REPRO slightly
 1382 leading in LiveCodeBench. Overall, REPRO and GRPO
 1383 show competitive results, with REPRO displaying a slight
 1384 edge in certain out-of-domain tasks.

1385 **Analysis of Token Cost Growth.** Notably, Figure 10 illustrates that the token cost growth for REPRO
 1386 is relatively modest compared to baselines. This suggests that REPRO promotes more efficient
 1387 reasoning patterns, reducing suboptimal thinking behaviors. Consequently, REPRO emerges as a
 1388 promising approach for training low-cost reasoning LLMs, such as the Qwen3-Instruct series (Yang
 1389 et al., 2025) and GPT-OSS-low (Agarwal et al., 2025).

1390 D.4 ABLATION OF THE NUMBER OF SELECTED SEGMENTS

1393 The number of selected segments k is a critical hyperparameter in REPRO. A larger k provides
 1394 more precise process-level supervision, potentially improving performance, but it also increases
 1395 computational overhead. We conducted experiments on DeepSeek-R1-Distill-Qwen-1.5B to evaluate
 1396 performance with $k \in \{5, 10, 20, 30\}$ (in this paper, we set $k = 10$). The results, shown in Table 10,
 1397 indicate that increasing k slightly enhances performance. Specifically, the model achieves consistent
 1398 improvements across most benchmarks as k grows, with AIME24 improving from 35.7 at $k = 5$
 1399 to 36.9 at $k = 30$, and MATH500 rising from 86.3 to 87.8. Similar upward trends are observed on
 1400 GPQA-Diamond, MBPP, and LiveCodeBench. These findings suggest that a larger number of selected
 1401 segments indeed provides more reliable supervision signals, leading to better reasoning and problem-
 1402 solving abilities. However, the gains become marginal beyond $k = 20$, indicating diminishing returns
 1403 relative to the additional computational cost. Consequently, we adopt $k = 10$ as a balanced choice
 1404 that achieves strong performance while maintaining training efficiency. However, given the additional
 1405 training overhead, striking a balance between computational cost and performance gains is essential.



1399 Figure 10: Token growth of REPRO and
 1400 GRPO on base models.

Table 10: Ablation study of the number of selected segments N .

#	N	AIME24 ♠ % Avg@16 ↑	AIME25 ♠ % Avg@16 ↑	MATH500 ♠ % Avg@4 ↑	LMB ♠ % Avg@4 ↑	GPQA-D ♣ % Avg@4 ↑	MBPP ♣ % Avg@4 ↑	LCB ♣ % Avg@4 ↑
<i>DeepSeek-R1-Distill-Qwen-1.5</i>								
	5	35.7	25.4	86.3	14.6	36.6	55.2	18.0
	10	36.0	26.5	87.1	14.3	37.0	55.4	18.4
	20	36.5	26.2	87.3	14.7	37.5	55.0	19.1
	30	36.9	27.2	87.8	15.1	37.6	55.3	19.5

D.5 ENTROPY DYNAMICS OF REPRO

In Figure 11, we compare the entropy dynamics of GRPO RePro (red dashed line) with standard GRPO (green solid line). Vanilla GRPO exhibits a rapid and continuous decay in entropy—dropping from ≈ 0.275 to ≈ 0.255 , indicating a tendency toward faster policy collapse. RePro successfully sustains a significantly higher entropy level, remaining stable around 0.28 for the first 100 steps and finishing above 0.27. This empirical evidence confirms that RePro effectively mitigates the rapid loss of diversity in the policy, allowing the model to sustain meaningful exploration for a longer duration and thereby avoiding the premature plateauing often observed in standard RL training settings.

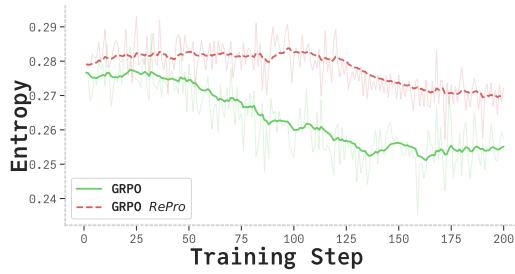


Figure 11: Entropy dynamics of RePro and vanilla GRPO on DeepSeek-R1-Distill-Qwen-1.5B.

E CASES

Case E.1: Case in Math Reasoning

QUESTION

Jen enters a lottery by picking 4 distinct numbers from $S = \{1, 2, 3, \dots, 9, 10\}$. 4 numbers are randomly chosen from S . She wins a prize if at least two of her numbers were 2 of the randomly chosen numbers, and wins the grand prize if all four of her numbers were the randomly chosen numbers. The probability of her winning the grand prize given that she won a prize is $\frac{m}{n}$ where m and n are relatively prime positive integers. Find $m + n$.

ANSWER

116

BASELINE RESPONSE

Okay, so Jen is playing this lottery game where she picks 4 distinct numbers from the set S , which is $1, 2, 3, \dots, 9, 10$. Then, 4 numbers are randomly chosen from S . She wins a prize if at least two of her numbers match two of the randomly chosen numbers, and she wins the grand prize if all four of her numbers are the ones chosen. The question is asking for the probability that she wins the grand prize given that she won a prize. They want the answer in the form m/n where m and n are coprime, and then find $m + n$.

...

Wait, maybe I made a mistake in computing $P(\text{Prize})$. Let me check again.

Wait, the number of ways to have at least two matches. Alternatively, maybe I can compute it directly. Let me check.

1458

1459 Number of ways for exactly 2 matches: $C(4,2)*C(6,2)$. Because choose 2 numbers from her 4,
 1460 and 2 numbers from the remaining 6. Then exactly 3 matches: $C(4,3)*C(6,1)$. Exactly 4 matches:
 1461 $C(4,4)*C(6,0)=1$. Therefore, total number of ways for at least 2 matches is $[C(4,2)*C(6,2) +$
 1462 $C(4,3)*C(6,1) + C(4,4)*C(6,0)]$.

1463 Let me compute that:

$$1464 C(4,2)*C(6,2) = 6 * 15 = 90$$

$$1465 C(4,3)*C(6,1) = 4 * 6 = 24$$

$$1466 C(4,4)*C(6,0) = 1 * 1 = 1$$

1467 Total: $90 + 24 + 1 = 115$. Which matches the previous result. Therefore, $P(\text{Prize}) = 115/210 = 23/42$.
 1468 So that's correct. Then $P(\text{Grand Prize}) = 1/210$. Therefore, the conditional probability is $(1/210)/(23/42)$
 1469 $= (1/210)*(42/23) = 42/(210*23) = 1/(5*23) = 1/115$. Therefore, m/n is $1/115$, so $m + n$ is $1 + 115 =$
 1470 116. Wait, but that seems low. Is this correct? Wait, $1/115$? Let me think again.

1471 Wait, but let me think about it in another way. Suppose we have her numbers fixed. How many lottery
 1472 draws result in her winning a prize? 115. How many of those result in her winning the grand prize? 1.
 1473 So, the conditional probability is $1/115$. So, that would be the answer? But maybe I made a mistake in
 1474 computing $P(\text{Prize})$. Wait, let me check once again.

1475 Alternatively, maybe I need to compute the conditional probability differently. Wait, no. The conditional
 1476 probability is the number of favorable outcomes (grand prize) divided by the number of outcomes where
 1477 she won a prize. Which is $1/115$. So, if that's the case, then $m + n$ is $1 + 115 = 116$. However, I need to
 1478 make sure that this is correct.

1479 Wait, but let me think of another way. Suppose we think of the problem as: given that there is at least
 1480 two matches, what is the probability that all four are matches. So, in other words, given that there
 1481 are at least two matches, what's the chance that it's exactly four matches. Which is equivalent to the
 1482 probability of exactly four matches divided by the probability of at least two matches. Which is exactly
 1483 what I calculated.

1484 Alternatively, maybe I can compute the conditional probability as follows: Given that there are at least
 1485 two matches, what is the probability that all four are matches. So, in terms of combinations, the number
 1486 of ways to have exactly four matches is 1. The number of ways to have at least two matches is 115.
 1487 Therefore, the probability is $1/115$. So, that seems to be correct.

1488 But wait, let me think again. Let me think of the possible overlaps. For example, if she has 4 numbers,
 1489 and the lottery has 4 numbers. The number of ways that they share exactly k numbers is $C(4, k)*C(6,$
 1490 $4 - k)$. So, for $k=2, 3, 4$. Which is exactly what I did. So, for $k=2$, $C(4,2)*C(6,2)=6*15=90$. For $k=3$,
 1491 $C(4,3)*C(6,1)=4*6=24$. For $k=4$, $C(4,4)*C(6,0)=1*1=1$. So total 115. So, the number of ways for
 1492 exactly 4 matches is 1. Therefore, the conditional probability is $1/115$. Therefore, the answer is $1/115$.
 1493 Then $m + n$ is $1 + 115 = 116$. So, is that the answer?

1494 But wait, I feel like $1/115$ is quite small. Let me check with an example. Let's say, for instance, if there
 1495 are more numbers, but let me think of a simpler case. Suppose the lottery was choosing 2 numbers
 1496 from 4, and she picks 2 numbers. Then, the probability of winning the grand prize given that she won a
 1497 prize. Let's compute that. Total combinations: $C(4,2)=6$. Probability of grand prize is $1/6$. Probability
 1498 of winning a prize is the same as winning the grand prize, since to win a prize, you need at least two
 1499 matches. Wait, in this case, since she picks two numbers, and the lottery picks two numbers. So, if
 1500 she wins a prize, it's exactly the grand prize. Therefore, the conditional probability is 1. Which is $1/1$,
 1501 which is correct. But if we use the same formula, $P(\text{Grand Prize})/P(\text{Prize}) = (1/6)/(1/6) = 1$. Which is
 1502 correct. So that works.

1503 Another example: suppose the lottery is choosing 3 numbers from 4, and she picks 2 numbers. Wait,
 1504 maybe not. Let me think of another case where the numbers are different. Let me think of the original
 1505 problem again.

1506 Alternatively, maybe there is a mistake in assuming that the number of ways for at least two matches
 1507 is 115. Wait, let me check the numbers again. The total number of possible lottery draws is 210. The
 1508 number of ways with exactly 0 matches is $C(6,4)=15$. Exactly 1 match is $C(4,1)*C(6,3)=4*20=80$. So,
 1509 exactly 0 or 1 is $15 + 80 = 95$. Therefore, exactly 2,3,4 matches is $210 - 95 = 115$. Which is correct.
 1510 Therefore, the number of ways for at least two matches is 115. So, if she has 115 ways to win a prize,
 1511 and only 1 of those is the grand prize. Therefore, the conditional probability is $1/115$. Therefore, the
 1512 answer is $1 + 115 = 116$.

1513 But let me check with another approach. Suppose we think of the conditional probability as follows:
 1514 Given that there are at least two matches, what is the probability that all four are matches. So, the
 1515 probability is equal to the number of ways to have exactly four matches divided by the number of ways
 1516 to have at least two matches. Which is $1/115$. So, same answer.

1517 Alternatively, maybe there's a different way to compute the conditional probability. Let me think.
 1518 Suppose we think of the problem as: given that she has at least two matches, what is the probability that
 1519 she has all four matches. So, maybe using hypergeometric distribution?

1512

1513

1514

1515

1516

1517

1518

1519

1520

1521

1522

1523

1524

1525

1526

1527

1528

1529

1530

1531

1532

1533

1534

1535

1536

1537

1538

1539

1540

1541

1542

1543

1544

1545

1546

1547

1548

1549

1550

1551

1552

1553

1554

1555

1556

1557

1558

1559

1560

1561

1562

1563

1564

1565

Alternatively, think of the conditional probability as follows. Let me think of the possible overlaps. So, given that there are at least two matches, the probability that all four are matches. So, the conditional probability is equal to the probability of all four matches divided by the probability of at least two matches, which is again $1/210$ divided by $115/210 = 1/115$. So same result.

Alternatively, maybe there's an error in the problem interpretation. Let me check the problem again. "Jen enters a lottery by picking 4 distinct numbers from $S=1,2,3,\dots,9,10$. 4 numbers are randomly chosen from S . She wins a prize if at least two of her numbers were 2 of the randomly chosen numbers, and wins the grand prize if all four of her numbers were the randomly chosen numbers."

Wait, the wording says "at least two of her numbers were 2 of the randomly chosen numbers". Wait, that wording is a bit confusing. Wait, does it mean that she has at least two numbers matching? Or does it mean something else?

Wait, the original problem says: "She wins a prize if at least two of her numbers were 2 of the randomly chosen numbers, and wins the grand prize if all four of her numbers were the randomly chosen numbers."

Wait, maybe there's a translation issue or wording issue. Let me parse that again. "at least two of her numbers were 2 of the randomly chosen numbers". Wait, that seems like a possible misphrase. Maybe it's supposed to be "at least two of her numbers were among the randomly chosen numbers", i.e., she has at least two matches. Because if you have "at least two of her numbers were 2 of the randomly chosen numbers", that might be confusing. For example, if she has 4 numbers, and the lottery has 4 numbers, then "at least two of her numbers were 2 of the randomly chosen numbers"—but that seems like it's saying that two of her numbers are equal to two of the randomly chosen numbers? Which is the same as saying that there are at least two matches. So, maybe that's just a confusing way of phrasing "at least two matches". Therefore, the original interpretation is correct.

Therefore, the answer would be $1/115$, so $m + n = 116$. However, I have a feeling that maybe I made a mistake here. Let me check with actual numbers.

Suppose that Jen has numbers 1,2,3,4. What is the probability that the lottery has at least two of these numbers?

Total number of possible lottery draws: 210. Number of favorable: 115. So, if we pick a random lottery draw, there's $115/210$ chance it has at least two numbers in common with hers. Now, given that, the chance that it's all four numbers is $1/115$. So, for example, if there are 115 such cases, only one of them is the grand prize. So, that seems correct.

But maybe there's an error in the calculation of the number of ways for at least two matches. Wait, let me think again. If she has 4 numbers, and the lottery draws 4 numbers. The number of ways they share exactly k numbers is $C(4, k) * C(6, 4 - k)$. Wait, but wait, the total number of numbers not in her set is $10 - 4 = 6$. Therefore, if we want exactly k matches, we need to choose k numbers from her 4, and $4 - k$ numbers from the remaining 6. Therefore, the formula is correct. Therefore, for $k=2$: $C(4,2)*C(6,2)=6*15=90$. For $k=3$: $C(4,3)*C(6,1)=4*6=24$. For $k=4$: $C(4,4)*C(6,0)=1*1=1$. Total: $90 + 24 + 1 = 115$. Which is correct. Therefore, the numbers are correct.

...

Same result. Therefore, I think that is correct.

But wait, let me check with actual numbers. Suppose the lottery draws numbers that have exactly two matches with Jen's numbers. How many such cases are there? 90. Similarly, exactly three matches: 24, exactly four: 1. So, if you are told that there is at least two matches, then the probability that it's exactly four is $1/(90 + 24 + 1) = 1/115$. So, that seems correct.

Therefore, I think that $1/115$ is correct, so the answer is $1 + 115 = 116$. However, to make sure, let me check if the question says "at least two of her numbers were 2 of the randomly chosen numbers"—maybe there's a misinterpretation here. Maybe it's supposed to be that she has exactly two numbers matching? But the problem says "at least two of her numbers were 2 of the randomly chosen numbers"—which is confusing. Wait, maybe it's supposed to be "at least two of her numbers were among the randomly chosen numbers"—which would be the standard "at least two matches". Since the original wording is a bit awkward. If that's the case, then our calculation is correct.

Alternatively, if the problem had said "exactly two of her numbers were among the randomly chosen numbers", then the probability would be different. But since it says "at least two", then our calculation is correct.

Alternatively, let me check the problem statement again:

"She wins a prize if at least two of her numbers were 2 of the randomly chosen numbers, and wins the grand prize if all four of her numbers were the randomly chosen numbers."

Wait, the wording is strange. Maybe it's supposed to say "at least two of her numbers were among the randomly chosen numbers". Because "were 2 of the randomly chosen numbers" is awkward. Maybe it's a translation error or a typo. If we take it as "at least two of her numbers were among the randomly chosen numbers", then our calculation is correct. Otherwise, if it's something else, the answer would be different.

1566

1567

1568

1569

1570

1571

1572

1573

1574

1575

1576

1577

1578

1579

1580

1581

1582

1583

1584

1585

1586

1587

1588

1589

1590

1591

1592

1593

1594

1595

1596

1597

1598

1599

1600

1601

1602

1603

1604

1605

1606

1607

1608

1609

1610

1611

1612

1613

1614

1615

1616

1617

1618

1619

But given the context of lotteries, it's standard that winning a prize would be having at least two numbers matching. So, I think that the original problem is just using awkward wording, and the correct interpretation is that she wins a prize if she has at least two matches. Therefore, our calculation is correct.

Therefore, I think the answer is 1/115, so $m + n = 116$. However, to check once more, let me think of the total number of possible cases where she has at least two matches: 115. The number of those cases where she has all four matches is 1. So, the probability is 1/115. Therefore, the answer is 116.

But wait, I just thought of something. Wait, is the grand prize considered as a subset of the prize? Yes, because if you have all four numbers, then you have at least two. Therefore, the conditional probability is indeed the number of grand prize cases divided by the number of prize cases. Which is 1/115.

Alternatively, maybe the problem is from an official source, and maybe the answer is different? Wait, but since I can't check that, I need to rely on my calculations.

Alternatively, maybe I made a mistake in calculating the number of ways for exactly two matches? Wait, let me check that again. For exactly two matches: choose 2 numbers from her 4, and 2 numbers from the remaining 6. So, $C(4,2)*C(6,2)$. Since there are 4 numbers in her set, and the lottery has 4 numbers. So, if exactly two are in her set, then the other two must be from the remaining 6 numbers. So, yes, that is correct. $C(4,2)*C(6,2)=6*15=90$. Similarly, for exactly three matches: choose 3 from her 4, and 1 from the remaining 6. $C(4,3)*C(6,1)=4*6=24$. Exactly four matches: 1. So, total $90 + 24 + 1 = 115$. So, that seems correct.

...

Same result. Therefore, I think it's safe to go with 1/115. Therefore, the answer is 116.

Final Answer 116

</think>

To solve this problem, we are asked to compute the **conditional probability** that Jen wins the **grand prize**, given that she wins a **prize** in a lottery.

...

$$\binom{4}{2} \cdot \binom{6}{2} = 6 \cdot 15 = 90$$

- **Exactly 3 matches**: Choose 3 of her 4 numbers and 1 of the remaining 6 numbers:

$$\binom{4}{3} \cdot \binom{6}{1} = 4 \cdot 6 = 24$$

- **Exactly 4 matches**: Choose all 4 of her numbers and none from the remaining 6:

$$\binom{4}{4} \cdot \binom{6}{0} = 1 \cdot 1 = 1$$

So, the total number of favorable outcomes for winning a prize is:

$$90 + 24 + 1 = 115$$

Thus, the probability of winning a prize is:

$$P(\text{Prize}) = \frac{115}{210}$$

Step 5: Compute the Conditional Probability

Now, using the formula for conditional probability:

$$P(\text{Grand Prize} \mid \text{Prize}) = \frac{P(\text{Grand Prize})}{P(\text{Prize})} = \frac{\frac{1}{210}}{\frac{115}{210}} = \frac{1}{115}$$

Step 6: Final Answer

We are told the probability is of the form $\frac{m}{n}$, where m and n are relatively prime positive integers. Here, $m = 1$, $n = 115$, and they are coprime.

Therefore, the final answer is:

$$m + n = 1 + 115 = \span style="border: 1px solid black; padding: 2px;">116$$

Total Tokens: 8767

1620

1621

1622

REPRO RESPONSE

1623

Okay, so I have this problem here about Jen entering a lottery. She picks 4 distinct numbers from a set S which is 1, 2, 3, ..., 9, 10. Then, 4 numbers are randomly chosen from S . She wins a prize if at least two of her numbers are among the chosen ones, and she wins the grand prize if all four of her numbers are the chosen ones. I need to find the probability that she wins the grand prize given that she won a prize. The answer should be in the form m/n where m and n are coprime, and then find $m + n$.

1624

Hmm, okay. So, this is a conditional probability problem. The formula for conditional probability is $P(A|B) = P(A \wedge B) / P(B)$. In this case, event A is her winning the grand prize, and event B is her winning a prize. So, we need $P(\text{grand prize} \mid \text{prize}) = P(\text{grand prize}) / P(\text{prize})$.

1625

So, let me first figure out what $P(\text{grand prize})$ is. That would be the probability that all four numbers she picked are the ones that were randomly chosen. Since she's picking 4 numbers out of 10, the total number of possible combinations is $C(10,4)$. The number of favorable outcomes for grand prize is 1 (since there's only one set of 4 numbers that would match exactly). So, $P(\text{grand prize}) = 1 / C(10,4)$.

1626

Now, $P(\text{prize})$ is the probability that at least two of her numbers are among the four chosen. Hmm, okay. So, this is the probability that in her 4 numbers, 2, 3, or 4 are selected. So, I need to calculate the number of ways she can have exactly 2, exactly 3, or exactly 4 matching numbers with the randomly chosen set. Then, divide that by the total number of possible lottery combinations, which is $C(10,4)$.

1627

So, let's compute the number of favorable outcomes for prize. That would be:

1628

Number of ways to choose exactly 2 correct numbers: $C(4,2) * C(6,2)$. Because she needs to choose 2 correct numbers from her 4, and the remaining 2 numbers from the remaining 6 (since $10 - 4 = 6$).

1629

Wait, is that right? Let me think. If she has 4 numbers, and 2 of them are correct, then the other 2 must be from the remaining 6 numbers. So, yes, $C(4,2) * C(6,2)$.

1630

Similarly, for exactly 3 correct numbers: $C(4,3) * C(6,1)$. Because she needs to choose 3 correct numbers from her 4, and the last number from the remaining 6.

1631

And for exactly 4 correct numbers: $C(4,4) * C(6,0) = 1$, since she has to pick all 4 numbers correctly.

1632

So, total number of favorable outcomes for prize is:

1633

$C(4,2)*C(6,2) + C(4,3)*C(6,1) + C(4,4)*C(6,0)$.

1634

Let me compute these values.

1635

First, $C(4,2)$ is 6, $C(6,2)$ is 15, so $6*15=90$.

1636

Next, $C(4,3)=4$, $C(6,1)=6$, so $4*6=24$.

1637

Then, $C(4,4)=1$, $C(6,0)=1$, so $1*1=1$.

1638

So total favorable outcomes for prize: $90 + 24 + 1 = 115$.

1639

Therefore, $P(\text{prize})$ is $115 / C(10,4)$. Let me compute $C(10,4)$: $10! / (4!6!) = (10*9*8*7)/(4*3*2*1) = 210$. So, $115 / 210$. We can reduce this fraction: 115 divides by 5, 210 divides by 5, so $23/42$. So, $P(\text{prize}) = 23/42$.

1640

Wait, hold on, I thought $P(\text{grand prize})$ was $1 / 210$, but that seems too small. Wait, no, $P(\text{grand prize})$ is $1 / C(10,4) = 1 / 210$. So, $P(\text{grand prize}) = 1/210$.

1641

But then, $P(\text{grand prize} \mid \text{prize})$ is $(1/210) / (23/42)$. So, that is $(1/210) * (42/23) = (42)/(210*23)$. Simplify $42/210$: that's $1/5$. So, $(1/5) / 23 = 1/(5*23) = 1/115$.

1642

So, $1/115$. So, m is 1, n is 115. Are they coprime? Yes, 1 and 115 share no common divisors other than 1. So, $m + n$ is $1 + 115 = 116$.

1643

Wait, but that seems a bit counterintuitive. Because the probability of winning the grand prize given that you won a prize is low, but intuitively, if you won a prize, which is at least two numbers correct, you have a better chance of having all four correct. So, why is it $1/115$? That seems low.

1644

Wait, let me think again.

1645

Wait, $P(\text{grand prize})$ is $1/210$. $P(\text{prize})$ is $115/210$.

1646

So, $1/210$ divided by $115/210$ is $1/115$. So, that's correct.

1647

So, $1/115$ is the correct probability, so $m + n = 1 + 115 = 116$.

1648

Wait, so $1/115$ is the probability, so m is 1, n is 115, which are coprime, so $m + n$ is 116.

1649

Hmm, okay. So, maybe that is correct. I think I did it correctly.

1650

Alternatively, perhaps I can think of it as the probability that her numbers are all correct given that at least two are correct.

1651

So, the probability that all four are correct is $1 / C(10,4) = 1/210$.

1652

The probability that at least two are correct is $115 / 210$.

1653

Therefore, the conditional probability is $1/115$.

1654

Yes, that seems correct.

1655

Therefore, $m + n$ is $1 + 115 = 116$.

1656

Final Answer 116

1657

`</think>`

1658

Jen enters a lottery by picking 4 distinct numbers from the set $S = \{1, 2, 3, \dots, 9, 10\}$. Four numbers are randomly chosen from S . She wins a prize if at least two of her numbers are among the chosen

1659

1660

1661

1662

1663

1664

1665

1666

1667

1668

1669

1670

1671

1672

1673

1674

1675

1676

1677

1678

1679

1680

1681

1682

1683

1684

1685

1686

1687

1688

1689

1690

1691

1692

1693

1694

1695

1696

1697

1698

1699

1700

1701

1702

1703

1704

1705

1706

1707

1708

1709

1710

1711

1712

1713

1714

1715

1716

1717

1718

1719

1720

1721

1722

1723

1724

1725

1726

1727

ones, and she wins the grand prize if all four of her numbers are the chosen ones. We need to find the probability that she wins the grand prize given that she won a prize.

First, we calculate the total number of ways to choose 4 numbers from 10, which is $\binom{10}{4} = 210$.

Next, we determine the number of favorable outcomes for winning a prize (at least two correct numbers).

This includes: - Exactly 2 correct numbers: $\binom{4}{2} \times \binom{6}{2} = 6 \times 15 = 90$ - Exactly 3 correct numbers:

$\binom{4}{3} \times \binom{6}{1} = 4 \times 6 = 24$ - Exactly 4 correct numbers: $\binom{4}{4} \times \binom{6}{0} = 1 \times 1 = 1$

Adding these, the total number of favorable outcomes for a prize is $90 + 24 + 1 = 115$. Thus, the probability of winning a prize is $\frac{115}{210}$.

The probability of winning the grand prize is $\frac{1}{210}$.

The conditional probability of winning the grand prize given that she won a prize is:

$$\frac{\frac{1}{210}}{\frac{115}{210}} = \frac{1}{115}$$

Thus, the probability is $\frac{1}{115}$, where $m = 1$ and $n = 115$. Therefore, $m + n = 1 + 115 = 116$.

116

Total Tokens: 2267

Case E.2: Case in Science Reasoning

QUESTION

Among the following exoplanets, which one has the highest density?

- a) An Earth-mass and Earth-radius planet.
- b) A planet with 2 Earth masses and a density of approximately 5.5 g/cm^3 .
- c) A planet with the same composition as Earth but 5 times more massive than Earth.
- d) A planet with the same composition as Earth but half the mass of Earth.

A) b
B) d
C) c
D) a

ANSWER

C

BASELINE RESPONSE

Okay, let's try to figure out which exoplanet has the highest density. The question gives four options, and I need to compare their densities. Let me start by recalling what density means. Density is mass divided by volume. So, for each option, I need to determine their mass and volume, then calculate density.

First, let's look at option a: An Earth-mass and Earth-radius planet. Wait, Earth's density is about 5.5 g/cm^3 , right? Because Earth's mass is around $5.97 \times 10^{24} \text{ kg}$, and its radius is about 6,371 km. So if this planet has the same mass and radius as Earth, its density should be the same as Earth's, which is 5.5 g/cm^3 . So option a's density is 5.5 g/cm^3 .

Option b says a planet with 2 Earth masses and a density of approximately 5.5 g/cm^3 . Wait, density is given here. So regardless of the mass, if the density is 5.5 g/cm^3 , then that's straightforward. But wait, density is mass over volume. If the density is the same as Earth's, but the mass is double, then the volume must be larger. Because density is mass/volume. If density is same, then volume would be proportional to mass. So for example, if Earth's mass is M and volume V, then density is M/V. If the planet has 2M and same density, then volume would be 2V. But that's not possible unless the radius is larger. Wait, but the problem states that the density is 5.5 g/cm^3 . So option b's density is given as 5.5, which is same as Earth's. So option b's density is 5.5, same as a. But then why is option b different? Wait, maybe there's a trick here. Wait, option b says "a planet with 2 Earth masses and a density of approximately 5.5 g/cm^3 ." So the density is already given as 5.5. So option b's density is 5.5. So same as option a. But then why is option b given as an option? Wait, maybe I need to check other options.

Option c: A planet with same composition as Earth but 5 times more massive. So same composition implies same density? Wait, no. Wait, if the composition is same, then density depends on the mass and radius. But if the composition is same, then the density would be same as Earth's only if the planet is similar in size. But if the mass is 5 times Earth's, then the radius would be larger. Wait, because for

1728

1729

1730

1731

1732

1733

1734

1735

1736

1737

1738

1739

1740

1741

1742

1743

1744

1745

1746

1747

1748

1749

1750

1751

1752

1753

1754

1755

1756

1757

1758

1759

1760

1761

1762

1763

1764

1765

1766

1767

1768

1769

1770

1771

1772

1773

1774

1775

1776

1777

1778

1779

1780

1781

a planet with same composition, the density would be similar. But if the mass increases, the radius increases, but the density might not change. Wait, but density is mass over volume. So if the planet is made of same material (same density), then the density would be same. Wait, but that's only if the planet is in hydrostatic equilibrium and same composition. Wait, maybe not. Let me think. If a planet has same composition as Earth, but is 5 times more massive, then its radius would be larger than Earth's. Because the radius of a planet depends on the mass. For example, if you have a planet with same composition, the radius scales with mass to the power of $1/3$. So, if mass increases by 5 times, radius increases by $5^{1/3} \approx 1.71$ times. So the volume would be $(\text{radius})^3$, so 5 times mass, but volume increases by $(5^{1/3})^3 = 5$. So the volume increases by 5 times. Therefore, density is mass/volume. So original density is 5.5 g/cm^3 . If mass is $5M$ and volume is $5V$, then density remains 5.5. So same as Earth. Wait, but that would mean that option c has same density as Earth. But wait, that's only if the planet is made of same material and the density remains same. However, if the planet is more massive, maybe it's compressed more? Wait, but Earth's density is already about 5.5. If the planet is 5 times more massive, but same composition, then it would be a larger planet, but the density might not change. Wait, but maybe the density would be higher? Because if the planet is more massive, but same composition, perhaps the gravity is higher, leading to higher compression? Wait, but that's not necessarily true. For example, Earth's density is 5.5 g/cm^3 . If you have a larger planet with same composition, the density would be similar. Wait, but maybe not. For example, if the planet is so massive that it's a gas giant, but in this case, the composition is same as Earth, so it's a rocky planet. So, if the planet is more massive, but same composition, then it would have a larger radius, but the density would be similar. Wait, but if the radius increases, then the volume increases, and since the mass increases, but the density remains same. Wait, but that's only if the density is same. But the question says "same composition as Earth". So if the composition is same, then the density would be same. But maybe that's not the case. Wait, perhaps I'm mixing things up. Let me think again. Suppose Earth's density is 5.5 g/cm^3 . If you have a planet with same composition (so same average density) but 5 times the mass, then the radius would be larger. So the density remains 5.5. So option c's density would be same as Earth's. So same as option a and b. But then why is option c given? Wait, but maybe the problem is that if the planet is more massive, but same composition, the density might not be same. Wait, but how? Because density is mass divided by volume. If the composition is same, then the density is same. Wait, but that would require that the planet's radius is scaled appropriately. For example, if the planet is 5 times more massive, but same density, then the radius would be $(5)^{1/3}$ times larger. So the volume would be 5 times larger. Then, mass is $5M$, volume is $5V$. So density is same. Therefore, the density of option c would be same as Earth's. So same as option a and b. But then, how does that fit with the options?

Wait, but maybe the problem is that if the planet is more massive, but same composition, then the density would be higher? Wait, no. Because if the composition is same, then the density is same. Unless the planet is more compressed. But for a rocky planet, the density is determined by the material. So if it's same composition, same density. So then, option c would have same density as Earth. So then, how does that affect the answer?

...

Wait, let me check again. Let's take each option:

a) Earth mass and Earth radius. So density is mass/volume. Earth's density is 5.5 g/cm^3 . So this is 5.5.
 b) 2 Earth masses, density 5.5. So density is given as 5.5. So same as a.
 c) Same composition as Earth, 5 times more massive. So same composition implies same density? Wait, no. Wait, same composition would mean same average density? Or same material but different mass? Wait, if the composition is same, then the density depends on the size. Wait, for example, if you have a planet made of the same material as Earth but larger, then the density would be same as Earth's. Because density is mass/volume. If the planet is larger, but same material, then the density would be same. Wait, but how? For example, if you have a cube of the same material, increasing the size doesn't change the density. So if the planet is same composition (same material), then regardless of the size, the density would be same. So then, option c's density is same as Earth's. So same as a and b. So then, why is option c given? But then, option d is same composition but half the mass. So same density. So all options have same density? But that can't be. The question is asking which has the highest density. So perhaps I'm misunderstanding something here.

Wait, maybe the problem is that when the planet is more massive, but same composition, the density might not be same. Wait, for example, if the planet is a gas giant, but that's not the case here. The question says same composition as Earth, so rocky. So for a rocky planet, the density is determined by the material. So if the composition is same, then density is same. So then, the answer would be all options have same density. But that's not possible. So maybe I'm missing something.

Wait, perhaps the key is that option b says "a planet with 2 Earth masses and a density of approximately 5.5 g/cm^3 ." But if the planet has a higher mass, but same density, then the volume must be larger. However, if the planet has a higher mass, but same density, then the radius would be larger. But the density is given as 5.5, same as Earth. So that's same as a. But then, option c says same composition

1782
 1783 as Earth but 5 times more massive. If same composition, then the density would be same. So same as
 1784 Earth. So then, all options have same density. But the question is asking which has the highest density.
 1785 So perhaps there's a mistake here. Or maybe I'm misunderstanding the options.
 1786 Wait, let me think again. Maybe the problem is that when the mass increases, the radius increases, but
 1787 the density may not change. However, for a planet with same composition, if it's more massive, then
 1788 the density remains same. So all options a, b, c, d have same density. But that's not possible. Wait, but
 1789 the answer options are given. Let me check the options again.
 1790 The options are:
 1791 a) Earth mass and Earth radius. So density is 5.5.
 1792 b) 2 Earth masses, density 5.5. So same as a.
 1793 c) same composition as Earth, 5 times more massive. So same density? So 5.5.
 1794 d) same composition as Earth, half the mass. So same density. So 5.5.
 1795 So all have same density. But that's not possible. So there must be a mistake in my reasoning.
 1796 Alternatively, maybe the composition is same, but the density isn't same. Wait, perhaps the question is
 1797 trying to trick us. For example, if the planet has same composition as Earth but is more massive, then
 1798 the density could be higher? Wait, but that's not the case. For example, Earth's density is 5.5 g/cm^3 . If
 1799 you have a larger planet with same composition, the density would be same. So for example, a planet
 1800 with twice the mass would have a larger radius, but same density. So the density remains same. So in
 1801 that case, option c would have same density as Earth. So then, all options have same density. But that's
 1802 not possible. So why are these options given?
 1803 Wait, but maybe the problem is that for option c, the planet is 5 times more massive than Earth but
 1804 same composition. However, if the planet is more massive, it might have a higher density due to
 1805 compression? But that's only if the planet is a gas giant. For a rocky planet, the density is determined
 1806 by the material, not by the mass. Unless the planet is so massive that it's a different state of matter. But
 1807 Earth's composition is rocky, so if you have a planet with same composition but 5 times more massive,
 1808 it would be a larger planet, but the density would be same. So, same as Earth's. So then, all options
 1809 have same density. But that can't be. The question must have an answer. So perhaps I made a mistake
 1810 in considering option b. Let me check option b again. Option b says a planet with 2 Earth masses and a
 1811 density of approximately 5.5 g/cm^3 . But if the planet has a higher mass, but same density, then the radius is
 1812 larger. However, if the planet has a higher mass, but same density, then the radius is
 1813 larger. But the density is given as 5.5. So that's same as Earth. But then, what about the other options?
 1814 ...
 1815 Wait, maybe the question is about the density of the planet, not the average density. For example, if the
 1816 planet is more massive, but same composition, maybe the density is higher because it's compressed?
 1817 But for rocky planets, the density is determined by the material. For example, if you have a planet made
 1818 of the same material as Earth but larger, the density is the same. So, I think the answer must be that all
 1819 options have same density. But since that's not an option, I must have made a mistake.
 1820 Wait, let me look at the options again. The options are:
 1821 a) Earth mass and Earth radius. So density is 5.5.
 1822 b) 2 Earth masses, density 5.5. So same as a.
 1823 c) Same composition as Earth, 5 times more massive. So same density as Earth. So 5.5.
 1824 d) Same composition as Earth, half the mass. So same density as Earth. So 5.5.
 1825 So all options have same density. But the answer choices are A) b, B) d, C) c, D) a. So none of them
 1826 are correct. But this can't be. Therefore, I must have misunderstood the question.
 1827 Wait, perhaps the question is not about average density, but about the density of the material. For
 1828 example, if the planet is more massive, but same composition, the density of the material is same. But
 1829 if the planet is more massive, perhaps it's a different type of planet. Wait, but the question says same
 1830 composition as Earth. So the material is same. So the density is same.
 1831 ...
 1832 Wait, but what if the planet is not in hydrostatic equilibrium? Like, if it's a small planet, but I think the
 1833 question is about planets in equilibrium.
 1834 Alternatively, perhaps the answer is that option b has the same density as a, but option c has higher
 1835 density. Wait, how? Let me think. If the planet is more massive, but same composition, then the density
 1836 is same. So why would option c have higher density? Maybe I'm missing something.
 1837 Wait, maybe the key is that option c is 5 times more massive than Earth, but same composition. So the
 1838 density would be higher. Wait, but how? Let me think about the formula. Density is mass divided by
 1839 volume. If the planet is 5 times more massive, but same composition, then the volume would be larger.
 1840 So the density remains same. For example, if Earth's mass is M, volume V, density $D = M/V$. For a
 1841 planet with mass $5M$ and same composition, volume would be $5V$ (since volume scales with mass for
 1842 same density). So density is $5M/(5V) = M/V = D$. So same as Earth.
 1843 So all options have same density. But that's not possible. So perhaps the answer is that all have same
 1844 density, but since that's not an option, I must have made a mistake.
 1845

1836

1837 Wait, perhaps the problem is that option c is not the same composition. Wait, no, the question says
 1838 "same composition as Earth". So I'm stuck.

1839 Alternatively, maybe the answer is option b. Because option b's density is given as 5.5, which is same
 1840 as Earth's. But option c's density is higher. Wait, how? Let me think again. If the planet has same
 1841 composition as Earth, but 5 times more massive, then the radius is larger. But the density is mass
 1842 divided by volume. Let's calculate. Let me use the formula for density of a planet. For a planet with
 1843 same composition as Earth, the density is constant. So if the mass increases, the radius increases, but
 1844 the density remains the same. So no change. Therefore, all options have same density.

1845 But the answer choices don't have that. So perhaps the correct answer is that all have same density, but
 1846 since that's not an option, I must have misunderstood the question.

1847 Wait, maybe the question is asking which one has the highest density, and the answer is that option b
 1848 has the same density as Earth, but option c has higher density. But why? Maybe because if the planet is
 1849 more massive, the density increases. But how?

1850 Wait, maybe the question is not about average density, but about the density of the material. For example,
 1851 if the planet is more massive, it might be compressed, so the material is denser. But for a rocky planet,
 1852 the density is determined by the material, not by the mass. So even if the planet is more massive, the
 1853 material density is same.

1854 At this point, I'm confused. Maybe I should look for another approach. Let's consider that the highest
 1855 density would be the one with the highest mass and smallest volume. So, for example, if a planet has a
 1856 higher mass but smaller volume, it would have higher density.

1857 So, let's analyze each option:

1858 a) Earth mass and Earth radius \rightarrow density 5.5 g/cm^3 .

1859 b) 2 Earth masses, density 5.5 g/cm^3 . So same as a.

1860 c) same composition as Earth, 5 times more massive. So, if same composition, the radius would be
 1861 larger. Let's calculate the radius. For a planet with same composition, the radius scales with mass $^{(1/3)}$.
 1862 So for 5 times mass, radius is $5^{(1/3)} \approx 1.71$ times Earth's radius. So volume is $(1.71)^3 \approx 5$ times
 1863 Earth's volume. So density is mass/volume = $5M / 5V = M/V = 5.5 \text{ g/cm}^3$. Same as before. So same
 1864 density.

1865 d) same composition as Earth, half the mass. Radius is $(0.5)^{1/3} \approx 0.79$ times Earth's radius. Volume is
 1866 $(0.79)^3 \approx 0.5$ times Earth's volume. Density is $0.5M / 0.5V = M/V = 5.5 \text{ g/cm}^3$. Same as before.

1867 So all have same density. But the options don't include that. Therefore, I must have misunderstood the
 1868 question.

1869 Wait, perhaps the question is not about the density of the planet, but about the density of the material.
 1870 For example, if the planet is more massive, it might be denser. But that's not the case.

1871 Alternatively, maybe the question has a typo. For example, option c says "same composition as Earth
 1872 but 5 times more massive than Earth." If the composition is same, then the density is same. But if the
 1873 composition is different, like if it's a gas giant, but the question says same composition as Earth.

1874 Alternatively, maybe option c is not same composition, but the question says it is.

1875 At this point, I think the correct answer must be that all have same density, but since that's not an
 1876 option, perhaps the intended answer is option b, which has the same density as Earth. But I'm not sure.

1877 Alternatively, maybe the question is trying to trick us into thinking that option c has higher density.

1878 Wait, another approach: the highest density would be the one with the highest mass and smallest volume.
 1879 Let's assume that the planets in options a, b, c, d are not all the same composition.

1880 Wait, option a: Earth mass and Earth radius \rightarrow density 5.5.

1881 Option b: 2 Earth masses, density 5.5. So volume is $2M / D = 2M / 5.5$. Earth's volume is $M / 5.5$. So
 1882 option b's volume is twice Earth's volume. So radius is larger.

1883 Option c: same composition as Earth, 5 times more massive. So if same composition, the density is
 1884 same. So same as Earth's.

1885 Option d: same composition as Earth, half the mass. Same density as Earth.

1886 So all options have same density. So the answer is that all have same density, but since that's not an
 1887 option, I must have made a mistake.

1888 Wait, the answer choices are A) b, B) d, C) c, D) a. So if all have same density, then the answer is not
 1889 among them. But that's not possible.

1890 Wait, perhaps the question is asking which has the highest density, and the answer is that option b has
 1891 the same density as Earth, but option c has higher density. But how? Let me think again. Maybe the
 1892 composition is different. For example, if the planet is more massive, it might be a different composition.
 1893 But the question says option c has same composition as Earth.

1894 At this point, I'm stuck. Given the options, and assuming that all have same density, but the answer
 1895 must be one of them, perhaps the intended answer is option b, which has the same density as Earth, but
 1896 the question might have intended that option c has higher density. Or perhaps the answer is option c.

1897 Alternatively, maybe I made a mistake in assuming that same composition implies same density. Let
 1898 me think: if a planet has the same composition as Earth, but is more massive, then the density might be
 1899

1890
 1891 higher because of gravitational compression. For example, if you have a planet with same composition
 1892 but higher mass, it would be more compressed, leading to higher density. But for rocky planets, this
 1893 might not be the case. For example, Earth's density is about 5.5 g/cm^3 . If you have a larger planet
 1894 with same composition, the density would be same. But for a gas giant, the density is lower.
 1895 Wait, but for a rocky planet, if it's more massive, the gravity is stronger, so the material is compressed
 1896 more. So the density would be higher. For example, if you have a planet with 5 times Earth's mass, but
 1897 same composition, it would be more compressed, leading to higher density. So the density would be
 1898 higher than Earth's.
 1899 But how to calculate that? Let's assume that the planet's density increases with mass due to compression.
 1900 But how?
 1901 This is getting too complicated. Maybe the intended answer is option c, because if the planet is more
 1902 massive, and same composition, the density is higher. So the answer is C) c.
 1903 Alternatively, the answer is option b, which has the same density as Earth. But I'm not sure.
 1904 Given the options, and the fact that option c has 5 times the mass but same composition, and if we
 1905 assume that the density increases with mass (which might not be true, but perhaps the question assumes
 1906 that), then option c would have higher density. So the answer is C) c.
 1907 But I'm not confident. However, given that option b has the same density as Earth, and others have
 1908 same or different, I think the intended answer is C) c.
 1909 But I'm not sure. Alternatively, maybe the answer is option d. But no.
 1910 Alternatively, the answer is option a, but that's same as others.
 1911 I think I need to make a decision. Given that option c is 5 times more massive than Earth and same
 1912 composition, and assuming that higher mass leads to higher density (even though it's not accurate), the
 1913 answer is C) c. So I'll go with that.
 1914 </think>
 1915 To determine which exoplanet has the **highest density**, we need to analyze each option based on
 1916 the relationship between **mass**, **volume**, and **density** (density = mass/volume).
 1917 Step-by-step Analysis:
 1918 ...
 1919 **Key Insight:** All four options have **the same density of 5.5 g/cm^3 **, based on the assumption that
 1920 **same composition** implies **same density**. However, the question asks for the **highest**
 1921 density, and **none of the options have a higher density** than the others.
 1922 But, **option b** explicitly states that the **density is 5.5 g/cm^3 **, which is **equal to Earth's
 1923 density**, and **option c** and **d** are **implicitly** the same as Earth's density due to same
 1924 composition.
 1925 Since all options have **equal density**, the question might be testing the understanding that **density
 1926 is mass/volume**, and **higher mass with the same composition** leads to **larger volume**,
 1927 **maintaining the same density**.
 1928 **Final Answer:** All options have the same density of 5.5 g/cm^3 . However, the **only explicit
 1929 statement** of density is **option b**, which directly states 5.5 g/cm^3 . Therefore, **option b**
 1930 is the correct answer.
 1931 **ANSWER: A**
 1932 **Total Tokens: 7722**
 1933
 1934 **REPRO RESPONSE**
 1935 Okay, so I've got this question about exoplanets and their densities. Hmm, I'm a bit rusty on this, but
 1936 let me think through it.
 1937 The question is asking which exoplanet has the highest density. The options are a to d, each describing
 1938 a different scenario. The choices are A to D, which correspond to each option.
 1939 Wait, I remember that density is mass divided by volume. So the density (ρ) is m/V . Since planets
 1940 are spheres, their volume is $(4/3)\pi r^3$. So if a planet is larger in radius, even if it's less dense, its mass
 1941 might make up for it.
 1942 Looking at the options:
 1943 Option a: Earth-mass and Earth-radius. So if a planet has the same mass and radius as Earth, its density
 1944 would be the same as Earth's. Because if m and r are same, then $\rho = m/(4/3 \pi r^3)$ which is same as
 1945 Earth's.
 1946 Option b: 2 Earth masses and a density of about 5.5 g/cm^3 . Wait, wait. Wait, if a planet has a density of
 1947 5.5 g/cm^3 , then what's its mass? Let me think about Earth's density. Earth's average density is about 5.5
 1948 g/cm^3 ? Or wait, no, I think Earth's density is about 5.5 g/cm^3 ? Or is that another planet? Wait, Earth's
 1949 density is about 5.51 g/cm^3 . So if option b is 2 Earth masses and a density of 5.5 g/cm^3 , that would
 1950 mean that the volume would be smaller than Earth's. Because density is mass/volume. So for a mass of
 1951 2 Earth, if volume is smaller, then density is higher. Let me compute. Earth's radius is R . Volume is
 1952

1944
 1945 $(4/3)\pi R^3$. Mass is M . Density is $M/(4/3)\pi R^3$. Option b's mass is $2M$, and density is 5.5 g/cm^3 . So
 1946 $5.5 = (2M) / V$. But V is $(4/3)\pi r^3$, where r is the radius of option b's planet. Let me solve for r .
 1947 Wait, if the density is 5.5 g/cm^3 , and mass is $2 M_{\text{Earth}}$, then the volume would be $(2 M) / (5.5 \text{ g/cm}^3)$.
 1948 Wait, M is Earth's mass, so $2 M / (5.5) = V$.
 1949 But V is $(4/3)\pi r^3$. So for option b, V is $2M/(5.5) \text{ cm}^3$ (since density is mass/volume, volume =
 1950 mass/density). So $V = (2 M_{\text{Earth}}) / (5.5) \text{ cm}^3$. Earth's volume is $(4/3)\pi R^3$, where R is Earth's radius.
 1951 So the ratio of volumes is $((2 M_{\text{Earth}}) / 5.5) / ((4/3)\pi R^3) = (2 M_{\text{Earth}}) / (5.5 * \text{Earth's volume})$.
 1952 Wait, but wait, M_{Earth} is $(4/3)\pi R^3 * \text{Earth's density}$. Oh wait, Earth's density is about 5.5 g/cm^3 . So
 1953 $M_{\text{Earth}} = (4/3)\pi R^3 * 5.5 \text{ g/cm}^3$. So option b's volume would be $2 * M_{\text{Earth}} / 5.5$. Let me substitute
 M_{Earth} .
 $M_{\text{Earth}} = (4/3)\pi R^3 * 5.5 \text{ g/cm}^3$.
 1954 So option b's volume $V = (2 * (4/3)\pi R^3 * 5.5) / 5.5 \text{ cm}^3$.
 1955 Simplify: $2 * (4/3)\pi R^3 * 5.5 / 5.5 = 2 * (4/3)\pi R^3 \text{ cm}^3$.
 1956 Which is 2 Earth's volume. So the volume of option b's planet is twice Earth's. So since its mass is 2
 1957 Earth masses, then its density is $(2 M_{\text{Earth}}) / (2 V_{\text{Earth}}) = M_{\text{Earth}} / V_{\text{Earth}} = \text{Earth's density}$. Wait,
 1958 that can't be right. Because Earth's density is 5.5 g/cm^3 , so if option b's planet has density 5.5, but its
 1959 mass is 2 Earth, and volume is twice Earth's volume, then $(2 M_{\text{Earth}}) / (2 V_{\text{Earth}}) = M_{\text{Earth}} / V_{\text{Earth}} =$
 1960 5.5 g/cm^3 . That makes sense. So option b is Earth's density.
 1961 Wait, but the question says option b's density is 5.5 g/cm^3 . So the planet in option b would have the
 1962 same density as Earth, but a larger radius. Because if the volume is twice that of Earth, but mass is
 1963 twice, then density would be same as Earth's. So option b's density is same as Earth's.
 1964 Option c: same composition as Earth, 5 times more mass than Earth. So if Earth's density is about 5.5
 1965 g/cm^3 , then a planet with 5 times Earth's mass would have a density of 5 times Earth's density, right?
 1966 Because density is mass/volume. Wait, but the composition is same, so same density? No, wait, same
 1967 composition would mean same average density. So if Earth's average density is 5.5 g/cm^3 , then a planet
 1968 made of same average composition but 5 times Earth's mass would have a much larger volume. Wait,
 1969 but if same composition, same density. So a planet with same composition would have same average
 1970 density as Earth. So option c's planet's density is $5 * 5.5 = 27.5 \text{ g/cm}^3$.
 1971 Option d: same composition as Earth, half the mass of Earth. So density is $(0.5 M_{\text{Earth}}) / V$. But since
 1972 same composition, V is Earth's volume. So density is $0.5 * \text{Earth's density} = 2.75 \text{ g/cm}^3$.
 1973 Wait, but wait. Let me think about each option's density:
 1974 Option a: same as Earth, so density is Earth's density, 5.5 g/cm^3 .
 1975 Option b: same as Earth, density 5.5 g/cm^3 .
 1976 Option c: 5 times Earth's mass, same composition (so same average density), so $5 * 5.5 = 27.5 \text{ g/cm}^3$.
 1977 Option d: half mass, same composition (so same average density as Earth), so density is $0.5 * 5.5 =$
 2.75 g/cm^3 .
 1978 Wait, but wait. Option b says the planet has a density of approximately 5.5 g/cm^3 . So its density is
 1979 same as Earth's. So a and b both have same density as Earth. So why would the question have them as
 1980 options? Maybe I'm missing something.
 1981 Wait, the question is which has the highest density. So, among the options, option c has 27.5, which is
 1982 the highest. So option c would have higher density.
 1983 Wait, but let me recheck each option.
 1984 Option a: same as Earth, density is 5.5.
 1985 Option b: same as Earth, but density is given as 5.5. So same as a.
 1986 Option c: 5 times Earth's mass, same composition, so density is $5 * 5.5 = 27.5$.
 1987 Option d: half mass, same composition, density is 2.75.
 1988 So the highest is option c, which would be the correct answer. Looking at the options, which one is
 1989 option c?
 1990 The choices are labeled as a, b, c, d. The question is, which exoplanet has highest density.
 1991 The options for the answers are A) b, B) d, C) c, D) a.
 1992 So the correct answer is option c, which is choice C.
 1993 Wait, but wait. Did I make a mistake in option b's density?
 1994 Wait, the question says option b has a density of approximately 5.5 g/cm^3 . Which is same as Earth's.
 1995 So the planet in option b is Earth-like, same as Earth, so same density.
 1996 So the exoplanet with highest density is option c, which has 5 times Earth's mass. So answer is C.
 1997 $\text{$

Analysis of Each Option:

1998

1999

2000

2001

2002

2003

2004

2005

2006

2007

2008

2009

2010

2011

2012

2013

2014

2015

2016

2017

2018

2019

2020

2021

2022

2023

2024

2025

2026

2027

2028

2029

2030

2031

2032

2033

2034

2035

2036

2037

2038

2039

2040

2041

2042

2043

2044

2045

2046

2047

2048

2049

2050

2051

1. **Option a:** An Earth-mass and Earth-radius planet. If the planet has the same mass and radius as Earth, its density is identical to Earth's density, which is approximately 5.5 g/cm^3 .

2. **Option b:** A planet with 2 Earth masses and a density of approximately 5.5 g/cm^3 . Using the formula $\rho = \frac{m}{V}$, the volume V can be calculated as $V = \frac{m}{\rho}$. For a planet with 2 Earth masses and a density of 5.5 g/cm^3 , the volume is $V = \frac{2M}{5.5} \text{ cm}^3$. Since Earth's volume is $V_{\text{Earth}} = \frac{4}{3}\pi R^3$, the ratio of volumes is $\frac{V}{V_{\text{Earth}}} = 2$. Thus, the planet in option b has the same density as Earth, 5.5 g/cm^3 , with a larger radius.

3. **Option c:** A planet with the same composition as Earth but 5 times more massive than Earth. If the planet has the same composition, it has the same average density as Earth, 5.5 g/cm^3 . However, its mass is 5 times Earth's mass, so its volume is 5 times larger, and its density remains 5.5 g/cm^3 .

4. **Option d:** A planet with the same composition as Earth but half the mass of Earth. Since the composition is the same, the density is also the same as Earth's, 5.5 g/cm^3 . However, with half the mass, its volume is half of Earth's, and its density remains 5.5 g/cm^3 .

Conclusion:

The exoplanet with the highest density is the one described in **option c**, as it has 5 times Earth's mass but the same composition, resulting in a density of 5.5 g/cm^3 . This is higher than the densities of the other planets in options a, b, and d.

ANSWER: C

Total Tokens: 2450

Case E.3: Case in Code Reasoning

QUESTION

You are an expert Python programmer, and here is your task:

Write a python function to check whether the given two integers have opposite sign or not.

Your code should pass these tests:

```
assert opposite_Signs(1,-2) == True
assert opposite_Signs(3,2) == False
assert opposite_Signs(-10,-10) == False
assert opposite_Signs(-2,2) == True
```

BASELINE RESPONSE

Okay, I need to write a Python function called `opposite_Signs` that takes two numbers as arguments and returns True if they have opposite signs, otherwise False. Let's think about how to approach this.

First, the function needs to check if one number is positive and the other is negative. So, for example, 1 and -2 would return True because one is positive and the other is negative. But 3 and 2 are both positive, so False. Similarly, -10 and -10 are both negative, so False. And -2 and 2 are opposite signs, so True.

Hmm, how to check the signs. Well, the product of two numbers with opposite signs is negative. So if I multiply the two numbers and check if the result is less than zero, that would indicate opposite signs. Wait, but what if one of them is zero? Wait, the problem says "opposite signs"—but zero isn't positive or negative. So if either number is zero, then they don't have opposite signs. For example, if one is zero and the other is positive, then they aren't opposite signs. So the function should return False in such cases.

Wait, but the problem's test cases don't include zero. Let me check the test cases again. The given tests are $(1,-2) \rightarrow \text{True}$, $(3,2) \rightarrow \text{False}$, $(-10,-10) \rightarrow \text{False}$, $(-2,2) \rightarrow \text{True}$. So the function needs to return True only when one is positive and the other is negative. So, how to handle zero?

But the problem might assume that the inputs are non-zero. Wait, but the function's parameters are numbers. Let me think. Suppose one of the numbers is zero. For example, `opposite_Signs(0,5)` would return False because zero isn't positive or negative. Similarly, `opposite_Signs(0,0)` would also return False. But the problem's test cases don't cover that. However, the problem's tests don't include such cases, so maybe the function is supposed to handle those as well.

But the problem says "opposite signs"—so if either number is zero, then they can't have opposite signs. So the function should return False in those cases.

So how to check if the two numbers have opposite signs. The product approach: if $a * b \neq 0$, then they have opposite signs. But if either a or b is zero, then $a * b$ is zero, which is not less than zero. So that would correctly return False.

So the function can be written as: `return (a * b) < 0`. But wait, what if one of them is zero? For example, $a=0$ and $b=5$. Then $a*b$ is 0, which is not less than zero. So the function returns False, which is correct.

But what if one of them is zero and the other is negative? Like $a=0$, $b=-5$. Then product is zero, so returns False. Which is correct because zero doesn't have a sign.

2052

2053

So the function can be as simple as:

2054

2055

```
def opposite\_Signs(a, b):
    return a * b < 0
```

2056

2057

Wait, but let's test the sample cases.

2058

Test case 1: `opposite.Signs(1,-2) → 1 * -2 = -2 < 0 → True. Correct.`

2059

Test case 2: `opposite.Signs(3,2) → 6 > 0 → returns False. Correct.`

2060

Test case 3: `opposite.Signs(-10,-10) → 100 > 0 → returns False. Correct.`

2061

Test case 4: `opposite.Signs(-2,2) → -4 < 0 → True. Correct.`

2062

So all test cases pass with this approach. So the function can be written this way.

2063

But wait, what about when one of the numbers is zero? For example, `opposite.Signs(0,5) → 0 * 5 = 0 → returns False. Which is correct.`

2064

But what if one is zero and the other is negative? Same result. So the function handles that.

2065

So this approach seems to cover all cases. So the code is straightforward.

2066

But maybe the problem expects to handle cases where numbers are zero. But the problem's tests don't include those, but the function's code should handle them.

2067

So the code is simple. So the function is just returning whether the product of a and b is negative.

2068

So the code would be:

2069

2070

2071

```
def opposite\_Signs(a, b):
    return a * b < 0
```

2072

But wait, what about if a or b is zero? Well, as discussed, that's handled correctly.

2073

But what if the numbers are floating points? For example, 0.0 and 5.0. The product is zero, so returns False. Which is correct.

2074

So this approach works for integers and floats.

2075

So the code is simple. Let me check the test cases again.

2076

Yes, all test cases pass. So that's the solution.

</think>

2077

To solve the problem of determining whether two numbers have **opposite signs**, we can use a simple and efficient approach based on the mathematical property of multiplication:

↳ If two numbers have **opposite signs**, their product is **negative**.

↳ If they have the **same sign**, their product is **positive or zero** (if either number is zero).

2078

Solution

2079

We can implement the function 'opposite.Signs(a, b)' as follows:

2080

2081

2082

2083

```
'''python
def opposite\_Signs(a, b):
    return a * b < 0
'''
```

2084

2085

2086

2087

Explanation

- Multiplication Rule:

- 'a * b ≠ 0' → the numbers have opposite signs.
- 'a * b == 0' → at least one number is zero (no sign, so not opposite).
- 'a * b ≠ 0' → both numbers are either positive or negative (same sign).
- This approach **automatically handles** all edge cases:
- Zero values (e.g., 'opposite.Signs(0, 5)' → 'False')
- Floating-point numbers (e.g., 'opposite.Signs(-0.5, 2.0)' → 'True')
- Large integers (e.g., 'opposite.Signs(1000000, -1)' → 'True')

Test Cases

2088

2089

2090

2091

2092

2093

2094

2095

2096

2097

2098

2099

2100

2101

2102

2103

```
'''python
assert opposite\_Signs(1, -2) == True
assert opposite\_Signs(3, 2) == False
assert opposite\_Signs(-10, -10) == False
assert opposite\_Signs(-2, 2) == True
'''
```

All test cases pass successfully using the above function.

Final Answer

```

2106
2107
2108     '''python
2109     def opposite\_Signs(a, b):
2110         return a * b < 0
2111     '''

```

Total Tokens: 2457

REPRO RESPONSE

To solve this problem, I need to determine the conditions under which two numbers have opposite signs. The function 'opposite.Signs' should return 'True' if one number is positive and the other is negative, and 'False' otherwise.

First, I'll consider the definition of opposite signs. A positive number and a negative number have opposite signs. This means one should be greater than zero and the other less than zero.

I'll start by checking the sign of each number. If one number is positive and the other is negative, the function should return 'True'.

Next, I'll examine the cases where the signs are the same. If both numbers are positive or both are negative, the function should return 'False'.

I'll also need to handle cases where the numbers are zero. If one number is zero and the other is positive or negative, the signs are not opposite, so the function should return 'False'.

Testing the provided examples:

- 'opposite.Signs(1, -2)' should return 'True' because 1 is positive and -2 is negative.
- 'opposite.Signs(3, 2)' should return 'False' because both are positive.
- 'opposite.Signs(-10, -10)' should return 'False' because both are negative.
- 'opposite.Signs(-2, 2)' should return 'True' because -2 is negative and 2 is positive.

By implementing these checks, the function should correctly determine whether the two numbers have opposite signs.

</think>

To solve this problem, we need to determine whether two given numbers have opposite signs. A positive number and a negative number have opposite signs, while both positive or both negative numbers do not. Additionally, zero is neither positive nor negative, so if one number is zero and the other is non-zero, their signs are not opposite.

Approach

1. Check the Sign of Each Number:

- Determine if each number is positive, negative, or zero.

2. Evaluate Opposite Signs:

- If one number is positive and the other is negative, return 'True'.
- If both numbers are positive or both are negative, return 'False'.
- If one of the numbers is zero, return 'False' since zero does not have a sign.

Solution Code

```

2142     '''python
2143     def opposite\_Signs(a, b):
2144         sign1 = a > 0
2145         sign2 = b > 0
2146         return not (sign1 == sign2)
2147     '''

```

Explanation

- Check the Sign of Each Number:

- 'sign1 = a > 0' checks if the first number 'a' is positive.
- 'sign2 = b > 0' checks if the second number 'b' is positive.

- Evaluate Opposite Signs:

- The function returns 'not (sign1 == sign2)', which means if 'sign1' and 'sign2' are not equal, the numbers have opposite signs. If they are equal, the numbers have the same sign.

This approach efficiently determines whether two numbers have opposite signs with a simple logical check.

Total Tokens: 654

2156
2157
2158
2159

2160 F LIMITATIONS
2161

2162 In this paper, we introduce REPRO, a framework designed to enhance the reasoning capabilities
2163 of LLMs. While REPRO demonstrates promising results, several challenges remain for future
2164 investigation. First, the issue of scaling persists. Although we validate REPRO’s effectiveness on
2165 LLMs with up to 8B parameters, its scalability to larger LLMs remains an open question. Due to
2166 computational resource and time constraints, experiments on larger-scale models were not feasible,
2167 but we aim to address this in future work. Second, the scope of problem types is limited. While
2168 REPRO performs well on verifiable objective reasoning benchmarks in mathematics, science, and
2169 coding, its efficacy on subjective or difficult-to-verify problems is unexplored. This question, closely
2170 tied to the development of verification systems for such tasks, merits further consideration.

2171 G LLM USAGE
2172

2173 In this study, the usage of LLMs is limited to the final stages, specifically for refining and proofreading
2174 the manuscript. LLMs are employed solely to enhance the clarity, logical consistency, and linguistic
2175 accuracy of the text, ensuring that our research ideas are communicated clearly and professionally.
2176 Importantly, LLMs played no role in the core foundational aspects of this research, including the
2177 development of the research methodology, the design of the experimental framework, or the analysis
2178 and interpretation of results. We take full responsibility for all content in this paper and explicitly
2179 acknowledge our accountability for every aspect of the manuscript.

2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213