

# REST: Stress Testing Large Reasoning Models by Asking Multiple Problems at Once

Zhuoshi Pan<sup>1,†,‡</sup>, Qizhi Pei<sup>2,3,‡</sup>, Yu Li<sup>2</sup>, Zinan Tang<sup>1</sup>, Qiyao Sun<sup>2</sup>,  
H. Vicky Zhao<sup>1,\*</sup>, Conghui He<sup>2</sup>, Lijun Wu<sup>2,\*</sup>

<sup>1</sup>Tsinghua University, <sup>2</sup>Shanghai Artificial Intelligence Laboratory, <sup>3</sup>Renmin University of China  
pzs23@mails.tsinghua.edu.cn, vzhao@tsinghua.edu.cn, wulijun@pjlab.org.cn

## Abstract

Recent Large Reasoning Models (LRMs) have achieved remarkable progress, yet their evaluation still relies on a narrow paradigm: evaluating one question at a time. This single-question setup suffers from two major limitations: (1) vulnerability to data contamination and diminishing difficulty, forcing costly creation of new questions with significant human effort, (2) failure to evaluate models under multi-context pressure, a key requirement for real-world deployment. To bridge this gap, we present **REST** (Reasoning Evaluation through Simultaneous Testing), a stress-testing framework that exposes LRMs to multiple problems simultaneously. Beyond basic reasoning, REST evaluates two under-tested capabilities: *contextual priority allocation* and *robustness against contextual interference*. Our evaluation of more than **30** advanced reasoning models on **9** reasoning benchmarks reveals several striking findings: Even state-of-the-art (SOTA) models such as *DeepSeek-R1* exhibit **substantial performance degradation under stress testing**, challenging the prevailing assumption that “LLMs are multi-problem solvers”. Crucially, **REST demonstrates stronger discriminative power** than existing benchmarks, revealing performance gaps among models that exhibit similar, near-ceiling performance under traditional evaluation. Some key insights emerge from our analysis: (1) the “**overthinking trap**” is a critical factor contributing to the performance degradation; (2) models trained with the “**Long2Short**” technique **preserve more of their single-problem accuracy** under REST, outperforming their standard-trained counterparts. These results establish REST as a cost-efficient, future-proof evaluation paradigm while reducing reliance on continuous human annotation. Code is available at <https://github.com/opedatalab/REST>.

\*Corresponding authors.

†Work during internship at Shanghai AI Lab.

‡Equal contribution.

## 1 Introduction

Recent years have witnessed remarkable progress in LRMs (Yang et al., 2024b; Guo et al., 2025; Du et al., 2025b; Pan et al., 2025), which exhibit strong performance across numerous reasoning tasks. Many benchmarks (Cobbe et al., 2021a; Hendrycks et al., 2021a; Rein et al., 2024; Du et al., 2025a) have been introduced to assess LRMs. However, in most existing benchmarks, questions are presented to LRMs one at a time, requiring the model to answer each independently. While this approach is effective for early model development, it now faces two fundamental challenges:

First, some widely used benchmarks, such as GSM8K (Cobbe et al., 2021a) and MATH (Hendrycks et al., 2021a), are rapidly losing their discriminative power, as many LRMs achieve near-saturated accuracy (e.g., *DeepSeek-R1* achieves 97.0% accuracy on MATH500). As a result, the community is forced into a costly cycle of benchmark obsolescence—continuously developing new, more difficult datasets (e.g., AIME25 (Ye et al., 2025)) while discarding still valuable ones. This raises our first key question: ***Can we extend the utility of existing benchmarks by increasing their difficulty, rather than constantly building new ones?***

Second, single-question evaluation fails to capture how models operate in real-world, multi-context scenarios requiring reasoning across multiple queries. For instance, in some conversation systems, large language models (LLMs) need to address multiple questions within a single run to reduce computational cost (Lin et al., 2024; Zhu et al., 2025). Also, in software development, LLMs must identify and fix multiple subtle bugs in a code snippet within a single structured output (Yang et al., 2025b). Similarly, legal compliance workflows require models to audit documents against dozens of independent criteria (e.g., liability caps, termi-

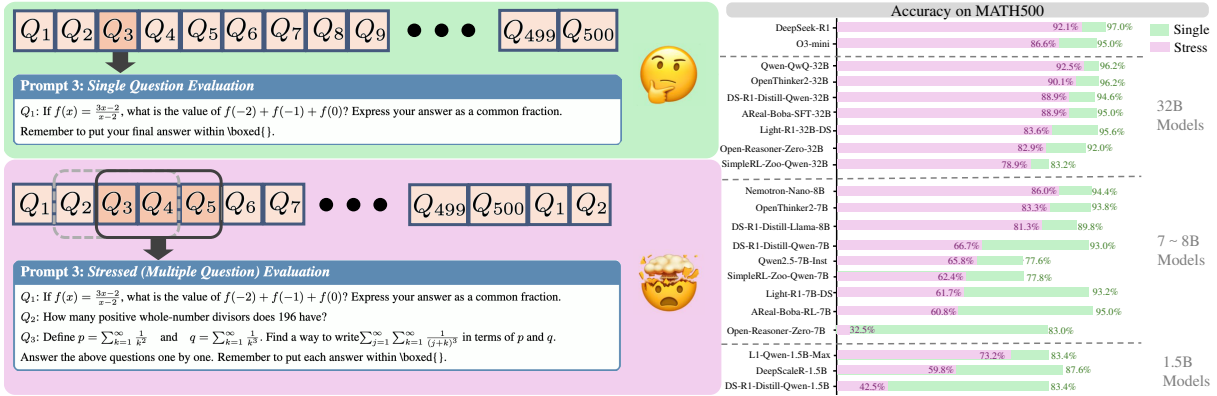


Figure 1: Illustration of REST evaluation. REST concatenates multiple questions into a single prompt, stress testing LLMs’ ability in handling increasing reasoning loads within a single reasoning process.

nation clauses) simultaneously (Trautmann et al., 2022). Although a few prior studies (Cheng et al., 2023; Lin et al., 2024; Wang et al., 2025c; Son et al., 2024) have investigated multi-question prompting, they only focus on simple tasks such as text classification (Semwal et al., 2018) or commonsense QA (Talmor et al., 2019), which are **insufficient** to assess the capabilities of current LLMs. This leads to our second question: **How well can current LLMs handle multiple reasoning-intensive questions simultaneously, and what factors affect their performance in such settings?**

To answer the aforementioned research questions, we propose **REST** (Reasoning Evaluation through Simultaneous Testing), an empirical and diagnostic framework that repurposes existing benchmarks into more challenging variants. Specifically, we transform **9** representative reasoning benchmarks (e.g., MATH500, AIME24 and GPQA) using REST, concatenating multiple questions into a single instruction to evaluate these questions at once (see Fig. 1 for illustration). Although this multi-query reasoning format may not fully match real-world tasks, it serves as a *controlled proxy* for stress testing reasoning models, and makes it possible to reveal weaknesses that remain hidden in conventional evaluation. Through a comprehensive evaluation of more than **30** LLMs, we have obtained several valuable findings:

- Even SOTA LLMs like DeepSeek-R1 (Guo et al., 2025) exhibit **significant performance degradation** under REST (e.g., 29.1% accuracy drop on AIME24), revealing a critical limitation in their reasoning robustness and challenging the common assumption that “LLMs are multi-problem solvers” (Cheng et al., 2023; Wang et al., 2025c; Son et al., 2024).

- REST offers **greater discriminative power**: models with near-identical accuracy on single-question evaluation settings diverge sharply when evaluated under REST.
- By question concatenation, REST effectively **revitalizes existing benchmarks**, making them challenging again for top-tier models.

Through further error analysis, we find that the “*overthinking phenomenon*” (Sui et al., 2025; Chen et al., 2025; Wang et al., 2025b)—where LLMs produce unnecessarily long and redundant reasoning—is a primary cause of performance degradation under REST. Moreover, LLMs post-trained with the “*Long2Short*” technique (Sui et al., 2025; Aggarwal and Welleck, 2025; Arora and Zanette, 2025), which encourages concise reasoning, perform better than their standard-trained counterparts under REST. These findings establish REST as a cost-effective and scalable alternative to constant benchmark replacement, while providing insights for building more robust reasoning systems capable of handling real-world, multi-query scenarios.

## 2 REST: Reasoning Evaluation through Simultaneous Testing

REST aims to enhance the utility of existing benchmarks by making them more challenging. In REST, we introduce a simple yet effective evaluation protocol that concatenates multiple questions into a single prompt to increase the reasoning load of LLMs. The simplicity of this evaluation protocol is intentional: by concatenating original questions rather than modifying them, REST avoids (1) the potential errors introduced by altering content and (2) the costly, expert-intensive process of creating new problems from scratch. The details of

benchmark reconstruction and evaluation of REST are presented in Sec. 2.1 and Sec. 2.2, respectively.

## 2.1 REST Benchmark Reconstruction

Let the original benchmark be denoted as  $\mathcal{Q} = \{q_1, q_2, \dots, q_N\}$ , where  $q_i$  represents an individual question and  $N$  is the total number of questions. REST transforms the benchmark into a new prompt set  $\mathcal{P}^s$  by concatenating  $s$  consecutive questions into each prompt. We refer to the parameter  $s \in \mathbb{Z}^+$  as the stress level, since a larger  $s$  imposes a greater reasoning burden on the model. Formally, for each  $i \in \{1, 2, \dots, N\}$ , we define the stress-level- $s$  prompt  $p_i^s$  as:

$$p_i^s = \text{Compose}(q_i, q_{i+1}, \dots, q_{[(i+s-1) \bmod N]}), \\ i \in \{1, 2, \dots, N\}.$$

The function  $\text{Compose}()$  formats multiple questions into a single prompt:

$$\text{Compose}(q_1, \dots, q_s) = Q_1 : \{q_1\}, \dots, Q_s : \{q_s\}.$$

Moreover, the subscript  $q_{[(i+s-1) \bmod N]}$  indicates cyclic indexing, which allows wrapping around when reaching the last question. By concatenating consecutive questions with cyclic indexing, each original question  $q_i$  appears exactly  $s$  times across all the prompts, and exactly once in each of the  $s$  possible positions within the concatenated prompts. This construction guarantees *fair question coverage* across different stress levels, ensuring that any observed performance degradation under REST arises from the increased reasoning load of handling multiple questions, rather than from artifacts of uneven question usage or positional bias.

After the transformation, REST produces a new prompt set  $\mathcal{P}^s = \{p_1^s, p_2^s, \dots, p_N^s\}$ , where each prompt contains  $s$  questions. Crucially, the transformed benchmark preserves the original size ( $|\mathcal{P}^s| = N$ ) while replacing single-question prompts with multi-question ones.

## 2.2 REST Evaluation

Given an LRM model  $f$ , we evaluate it on each prompt  $p_i^s$  to obtain an output response  $o_i^s = f(p_i^s)$ . From this response, we extract the individual predicted answers  $\{\hat{a}_i^s, \hat{a}_{i+1}^s, \dots, \hat{a}_{[(i+s-1) \bmod N]}^s\}$  using a function  $\text{Extract}()$ :

$$\{\hat{a}_i^s, \hat{a}_{i+1}^s, \dots, \hat{a}_{[(i+s-1) \bmod N]}^s\} = \text{Extract}(o_i^s).$$

To facilitate accurate extraction, we instruct LRMs to format their answers in task-specific ways.

For Mathematical and GPQA problems, we instruct the model to put each answer within “`\boxed{\}`”, while for code generation tasks, answers must be wrapped in Python code blocks “````python`”. Notably, although a more structured formatting method like JSON output could offer convenience for output parsing, we deliberately avoid imposing such strict formatting constraints, as prior studies (Tam et al., 2024; Shorten et al., 2024) indicate that such requirements can degrade performance.

We employ both rule-based and LLM-based extraction methods: (a) The rule-based method uses regular expressions to extract answers from predefined markers (e.g., “`\boxed{\}`”); (b) The LLM-based method prompts a model to retrieve the predicted answer from the response for each question. Further implementation details can be found in Appendix G. Finally, we define the model’s accuracy at stress level  $s$  by comparing the predicted answer  $\hat{a}$  and the ground truth answer  $a$ :

$$\text{Acc}(\mathcal{P}^s) = \frac{1}{N} \sum_{i=1}^N \text{Acc}(p_i^s) = \frac{1}{N} \sum_{i=1}^N \\ \frac{1}{s} \sum_{j=1}^s \delta(\hat{a}_{[(i+j-1) \bmod N]}^s, a_{[(i+j-1) \bmod N]}^s),$$

where  $\delta$  is the Kronecker delta function such that  $\delta(i, j) = 1$  if and only if  $i = j$  and 0 otherwise.

## 3 Experiment

### 3.1 Evaluation Setup

We evaluate over 30 LRMs, spanning a parameter size range from 1.5B to 671B. The temperature and top-p parameters are configured according to the corresponding official guidelines for each model. We set a maximum output token length of 32K for reasoning models and 8K for non-reasoning models. In addition, we conduct experiments with an extended 128K token limit, but observe negligible performance differences, as shown in Tab. 14 of Appendix M. Our evaluation is based on OpenCompass\* toolkit. To ensure consistency, we adopt the official prompt for each task, with a minor format requirement for multi-question responses: “Answer the above questions one by one. Remember to put each answer within `\boxed{\}` (or ````python` for code generation).” We further validate our findings with alternative prompt designs and different sampling parameters (Appendix D and E), confirming that results are robust and not artifacts.

\*<https://github.com/open-compass/opencompass>

Table 1: Evaluation results on various math benchmarks. The ‘‘Stress’’ column reports the average performance across four stress levels, as described in Sec. 3.1. Models that perform best under ‘‘Stress’’ are highlighted in gray. Results on non-math tasks are in Tab. 2 and Tab. 10 (Appendix); additional models are in Tab. 12 (Appendix).

Bench Model	GSM8K		MATH500		AMC23		AIME24		AIME25		AVG.	
	Single	Stress	Single	Stress	Single	Stress	Single	Stress	Single	Stress	Single	Stress
<i>1.5B LRMs</i>												
DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025)	84.62	70.21	83.40	42.47	62.50	13.98	29.17	4.97	25.00	5.91	56.94	27.51
DeepScaleR-1.5B (Luo et al., 2025)	84.84	66.58	<b>87.60</b>	59.77	76.25	32.05	<b>38.75</b>	12.82	<b>31.25</b>	14.23	<b>63.74</b>	37.09
L1-Qwen-1.5B-Max (Aggarwal and Welleck, 2025)	84.17	<b>78.29</b>	83.40	<b>73.23</b>	<b>77.50</b>	<b>48.37</b>	20.00	<b>15.13</b>	22.92	<b>14.95</b>	57.60	<b>45.99</b>
Qwen2.5-Math-1.5B-Inst (Yang et al., 2024c)	<b>85.37</b>	67.49	73.00	53.94	57.50	22.22	10.83	6.17	10.83	2.83	47.51	30.53
<i>7 ~ 8B LRMs</i>												
DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025)	89.49	89.06	93.00	66.75	87.50	36.06	54.17	16.53	35.42	11.37	71.92	43.95
Efficient-R1-7B ( $\alpha = 0.2$ ) (Arora and Zanette, 2025)	87.95	80.38	88.20	76.41	85.00	48.05	50.42	22.12	33.75	17.25	69.06	48.84
Nemotron-Nano-8B (Bercovich et al., 2025)	91.36	70.52	94.40	<b>86.04</b>	90.00	<b>76.24</b>	63.33	43.55	50.00	<b>32.28</b>	77.82	<b>61.71</b>
AReal-boba-RL-7B (Mei et al., 2025)	91.66	77.80	95.00	60.77	91.25	32.94	61.25	21.43	45.83	12.33	77.00	41.05
Light-R1-7B-DS (Wen et al., 2025)	88.05	82.69	93.20	61.73	90.00	34.91	55.83	16.63	45.83	12.96	74.58	41.78
OpenR1-Qwen-7B (Face, 2025)	<b>95.60</b>	90.22	92.20	81.64	83.75	54.11	47.50	26.77	32.92	21.19	70.39	54.79
OpenThinker2-7B (Team, 2025c)	94.39	91.99	93.80	83.30	85.00	63.23	54.58	34.50	41.67	23.66	73.89	59.33
Qwen2.5-Math-7B-Inst (Yang et al., 2024b)	95.53	78.53	83.60	56.59	60.00	28.46	14.17	6.40	11.67	5.33	52.99	35.06
Qwen2.5-7B-Inst (Yang et al., 2024a)	92.27	<b>85.12</b>	77.60	65.78	42.50	34.46	10.00	7.02	3.75	3.32	45.22	39.14
Qwen3-8B (Yang et al., 2025a)	95.30	<b>93.90</b>	<b>95.40</b>	80.28	<b>92.50</b>	53.59	<b>74.58</b>	<b>44.28</b>	<b>63.33</b>	32.06	<b>84.22</b>	60.82
<i>32B LRMs</i>												
DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025)	95.54	95.50	94.60	88.97	94.75	<b>86.24</b>	72.92	52.51	51.67	33.83	81.90	71.41
Qwen-QwQ-32B (Team, 2025d)	95.83	<b>95.78</b>	96.20	<b>92.49</b>	95.00	82.89	78.75	54.79	69.58	41.53	87.07	<b>73.49</b>
AReal-boba-SFT-32B (Mei et al., 2025)	95.01	94.75	95.00	88.92	<b>97.50</b>	78.96	77.50	45.79	60.00	33.55	85.00	68.39
Light-R1-32B-DS (Wen et al., 2025)	95.83	94.79	95.60	83.66	96.25	68.80	77.50	41.26	60.00	33.80	85.04	64.46
Qwen2.5-32B-Inst (Yang et al., 2024a)	95.53	93.77	82.20	73.39	60.00	49.72	20.00	9.61	16.67	6.73	54.88	46.64
Qwen3-32B (Yang et al., 2025a)	<b>95.91</b>	93.81	<b>96.60</b>	83.64	93.75	80.62	<b>85.00</b>	<b>56.67</b>	<b>71.67</b>	<b>44.53</b>	<b>88.59</b>	71.85
<i>MoE LRMs</i>												
GPT-OSS-20B (OpenAI, 2025a)	88.02	49.31	88.40	46.96	97.50	56.56	66.25	36.20	63.33	29.87	80.70	43.78
GPT-OSS-120B (OpenAI, 2025a)	84.91	47.65	89.60	60.06	93.75	71.93	67.08	<b>60.28</b>	68.33	<b>52.39</b>	80.73	58.46
Qwen3-30B-A3B (Yang et al., 2025a)	96.13	94.91	96.40	80.36	<b>98.75</b>	72.34	79.58	48.38	<b>68.75</b>	28.67	<b>87.92</b>	64.93
DeepSeek-R1 (Guo et al., 2025)	<b>96.20</b>	<b>96.16</b>	<b>97.00</b>	<b>92.09</b>	93.75	<b>81.80</b>	<b>81.66</b>	52.49	<b>68.75</b>	37.17	87.47	<b>71.94</b>
<i>Proprietary LRMs</i>												
O3-mini (OpenAI, 2025b)	95.83	<b>93.85</b>	95.00	86.62	90.00	59.17	79.16	34.07	71.66	20.63	86.25	55.51
O4-mini (OpenAI, 2025b)	93.71	93.07	90.00	82.40	96.25	<b>82.79</b>	73.33	49.69	80.00	41.42	<b>86.66</b>	69.87
Gemini-2.5-Flash-Thinking (Team et al., 2023)	89.23	91.28	<b>97.20</b>	69.92	<b>97.50</b>	47.63	76.67	26.60	71.67	16.54	86.45	50.39

We select 9 representative benchmarks for evaluation, and in REST, we set different stress levels  $s$  for each benchmark. Specifically, for relatively simple benchmarks like GSM8K, ARC-C and BBH, the stress levels are  $s \in \{1, 3, 6, 9, 12\}$ ; for medium-difficulty benchmarks, including MATH500 and AMC23, they are set as  $s \in \{1, 3, 5, 7, 9\}$ ; and for more challenging benchmarks, such as AIME24, AIME25, GPQA-Diamond, and LiveCodeBench (v5), the stress levels are  $s \in \{1, 2, 3, 4, 5\}$ . For answer extraction, we use the rule-based method by default, as we find no significant difference compared to the LLM-based method (see Sec. G in Appendix). To ensure clarity and consistency in evaluation, we report the performance of REST as the **average accuracy** across stress levels greater than 1. For relatively small benchmarks like AIME24, AIME25, and AMC23, we conduct 8 random runs and report the average to reduce variance. Results on ARC-C and BBH are included in Tab. 10 (Appendix).

### 3.2 Evaluation Result

Tab. 1, Tab. 2 and Tab. 10 (Appendix) present the performance of various LRMs on the original reasoning benchmarks (Single) and their transformed counterparts by REST (Stress). Our key findings can be summarized as follows:

#### LRMs can handle multiple simple questions at once, but struggle with challenging ones.

LRMs can address several relatively simple problems within a single run. As shown in Tab. 1, on GSM8K, the accuracy drop under REST for DeepSeek-R1-Distill-Qwen-7B (R1-7B) is just 0.43%. However, *this robustness diminishes rapidly on more challenging problems*. As shown in Tab. 1, even DeepSeek-R1 suffers substantial performance degradation (e.g., 31.58% accuracy drop on AIME25) when asked to solve multiple Olympiad-level problems within a single reasoning process. This performance gap is even more pronounced for smaller models, such as R1-7B,

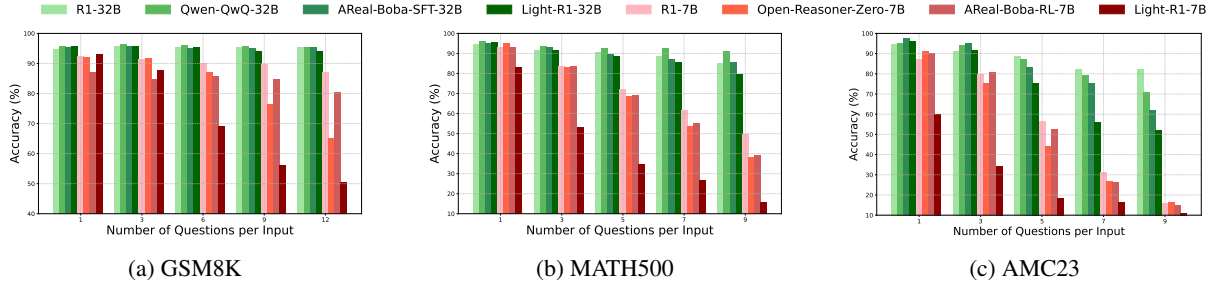


Figure 2: Performance comparison of LLMs of different sizes under various stress levels. REST reveals significant performance disparities among models of different sizes, although they demonstrate comparable, near-ceiling performance in traditional single-question evaluations.

whose accuracy on AIME24 drops sharply from 54.17% to 16.53% under REST. This contrast—strong multi-question performance on simple tasks but sharp degradation on hard ones—indicates that the performance drop under REST stems not from format artifacts, but from inherent limitations in the LLMs’ reasoning capacity. To further investigate the reasons, we provide quantitative analysis in Sec. 4 and case studies in Sec. C (Appendix).

**REST enhances the discriminative power of existing benchmarks.** By concatenating multiple problems into a single prompt, *REST amplifies performance gaps among models that exhibit comparable, near-ceiling accuracy* in traditional single-question evaluations. For example, on MATH500, DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-32B achieve single-question accuracies of 93.0% and 94.6%, respectively, with R1-7B trailing by just 1.6%. However, the accuracy of R1-7B plummets to 66.75% under REST, while R1-32B maintains a higher 88.97%, revealing a pronounced 22.22% performance gap. This contrast is similarly evident among models of the same size (e.g., AReal-boba-RL-7B and OpenThinker2-7B). Similar trends are also observed on AMC23, validating that REST can significantly enhance the discriminative power of existing medium-difficulty benchmarks through problem concatenation.

**Limitations of downstream post training.** The success of DeepSeek-R1 (Guo et al., 2025) has inspired numerous follow-up studies (Luo et al., 2025; Mei et al., 2025; Wen et al., 2025), which fine-tune DeepSeek distilled models on downstream tasks (e.g., math and code). Although such downstream post training yields improved single-question accuracy, our findings suggest that *these models cannot maintain their performance advantage in multi-question scenarios as in REST*.

Table 2: Evaluation results on GPQA-Diamond and LiveCodeBench (v5). The “Stress” column reports the average performance across four distinct stress levels, as described in Sec. 3.1. Models that perform best under “Stress” are highlighted in gray. Results on additional models are reported in Tab. 11 (Appendix).

Model	Bench		LiveCodeBench (v5)	
	GPQA-Diamond Single	Stress	Single	Stress
<i>1.5B LLMs</i>				
DeepSeek-R1-Distill-Qwen-1.5B	<b>37.37</b>	22.11	15.05	0.48
DeepScaleR-1.5B	31.82	27.90	<b>21.15</b>	1.83
L1-Qwen-1.5B-Max	36.87	<b>32.03</b>	19.35	<b>2.45</b>
Qwen2.5-1.5B-Inst	26.26	21.52	1.79	0.37
<i>7 ~ 8B LLMs</i>				
DeepSeek-R1-Distill-Qwen-7B	51.01	31.67	37.63	2.89
Efficient-R1-7B ( $\alpha = 0.2$ )	47.97	34.37	38.71	2.50
Nemotron-Nano-8B	51.01	34.06	50.90	8.56
AReal-boba-RL-7B	48.98	29.13	37.99	4.73
Light-R1-7B-DS	41.91	30.32	39.07	2.85
OpenR1-Qwen-7B	38.38	36.04	4.66	1.27
OpenThinker2-7B	49.49	40.60	39.43	14.51
Qwen2.5-7B-Inst	35.86	35.15	13.98	10.19
Qwen3-8B	<b>61.11</b>	<b>58.60</b>	<b>56.99</b>	<b>31.13</b>
<i>32B LLMs</i>				
DeepSeek-R1-Distill-Qwen-32B	60.10	53.73	55.56	26.71
Qwen-QwQ-32B	63.64	<b>60.03</b>	62.37	<b>32.16</b>
AReal-boba-SFT-32B	63.13	50.59	60.93	31.74
Light-R1-32B-DS	<b>65.66</b>	50.11	60.93	28.22
Qwen2.5-32B-Inst	42.93	40.04	26.88	24.00
Qwen3-32B	63.33	48.13	<b>62.72</b>	30.98
<i>MoE LLMs</i>				
GPT-OSS-20B	53.54	59.52	43.37	29.83
GPT-OSS-120B	60.10	<b>65.93</b>	63.08	<b>49.16</b>
Qwen3-30B-A3B	64.65	50.74	61.65	38.48
DeepSeek-R1	<b>70.20</b>	64.63	<b>63.44</b>	40.83
<i>Proprietary LLMs</i>				
O3-mini	71.21	67.39	60.21	48.36
O4-mini	76.26	<b>73.11</b>	<b>70.61</b>	<b>63.07</b>
Gemini-2.5-Flash-Thinking	<b>78.79</b>	68.00	61.65	48.34

For instance, models like AReal-boba-RL-7B and Light-R1-7B-DS, both post trained on R1-7B, achieve single-question accuracies of 95.00% and 93.20% on MATH500, slightly higher than the 93.00% accuracy of R1-7B. However, under REST, their accuracies drop significantly to 60.77% and 61.73%, respectively, both falling behind the

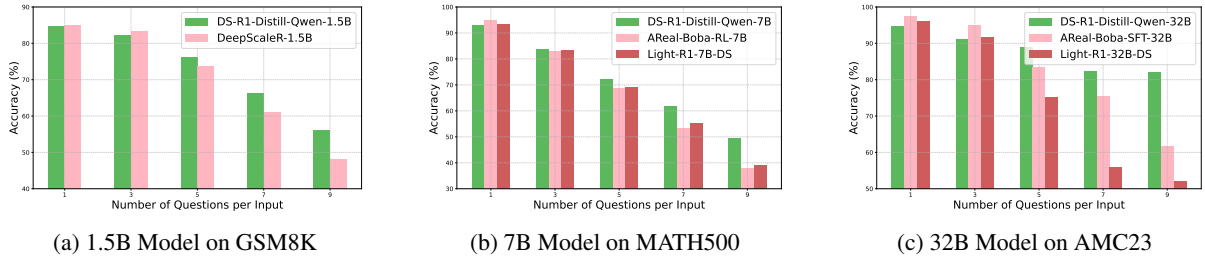


Figure 3: Limitation of downstream post training. Downstream post training exacerbates performance degradation under higher stress levels compared to the original DeepSeek distilled models.

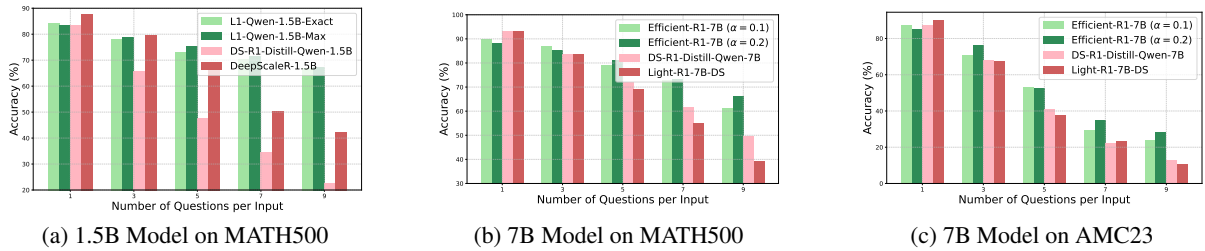


Figure 4: The effect of Long2Short training on REST. Long2Short training mitigates the performance degradation under higher stress levels.

66.75% achieved by R1-7B. This discrepancy becomes even more pronounced at higher stress levels (e.g., stress level  $s = 9$ ), as detailed in Fig. 3b. The performance degradation also extends to larger models, such as Light-R1-32B-DS and R1-32B, and is consistently observed across other benchmarks like AMC23, as shown in Tab. 1.

**Potential of “Long2Short” Training.** Despite their impressive performance, LRMs often produce unnecessarily lengthy reasoning traces (Sui et al., 2025; Chen et al., 2025). To mitigate this issue, recent studies (Aggarwal and Welleck, 2025; Arora and Zanette, 2025; Team et al., 2025b) propose the “Long2Short” technique, which introduces a length reward in the training objective to promote more concise and efficient reasoning. Our experiments reveal that “Long2Short” training also enhances reasoning robustness under REST. As illustrated in Fig. 4, models trained with “Long2Short”, such as L1-Qwen-1.5B-Max (Aggarwal and Welleck, 2025), exhibit notable performance advantages at high stress levels. For instance, as shown in Tab. 19 (Appendix), L1-Qwen-1.5B-Max surpasses R1-1.5B by a substantial 44.71% accuracy margin at stress level  $s = 9$  on MATH500. A similar trend is also observed among 7B models (e.g., Efficient-R1-7B ( $\alpha = 0.2$ ) vs. R1-7B). Further error analysis (Sec. Q in Appendix) confirms that “Long2Short” mitigates not only length-related failures but also reasoning degradation patterns, enhancing LRM’s resilience in multi-question settings.

## 4 Further Analysis

**Performance drops are NOT primarily due to output length limits.** To uncover the underlying reasons for LRMs’ failures in REST, we analyze the primary factors contributing to performance degradation under multi-question stress. A natural hypothesis is that models fail because they exceed the maximum output length when solving multiple problems. However, our error analysis reveals that this is not the primary cause—*models often fail even when their responses remain within the maximum output token limit.*

To investigate the cause of errors, we categorize model failures into six types (Tab. 3) and manually analyze the error types across four representative LRMs on AIME24 (Fig. 5). Additional error analysis on more models and benchmarks is in Tab. 16 (Appendix). For DeepSeek-R1 (Fig. 5d), Question Omission (QO) and Reasoning Error (RE) dominate, together accounting for 42.00% of failures. The high frequency of Question Omission (QO) suggests that DeepSeek-R1 suffers from a pronounced *question position bias*, focusing more on the first question in multi-problem prompts, despite explicit instructions to address each question sequentially. In contrast, error causes like Summary Error (SE), Output Truncation (OT), and Endless Repetition (ER) are rare in such large models.

However, for smaller models like Light-R1-7B (in Fig. 5b), the *error profile shifts* significantly, with Endless Repetition (ER) and Output Trunca-

Table 3: Six common causes of the performance degradation in REST.

Error Cause	Definition
<b>Output Truncation (OT)</b>	The output is truncated due to reaching the maximum output length (32K tokens for LLMs).
<b>Endless Repetition (ER)</b>	The model repeatedly generates the same phrase or sentence in its output.
<b>Format Violation (FV)</b>	The model fails to comply with the required output format (e.g., not putting the answer within “boxed{ }”).
<b>Question Omission (QO)</b>	The model fails to respond to all the given questions in the thinking process. (e.g., only answers the first question)
<b>Summary Error (SE)</b>	The model fails to summarize all the answers generated in the thinking process.
<b>Reasoning Error (RE)</b>	Error occurs in the reasoning process, such as calculation mistakes or concept misunderstandings.

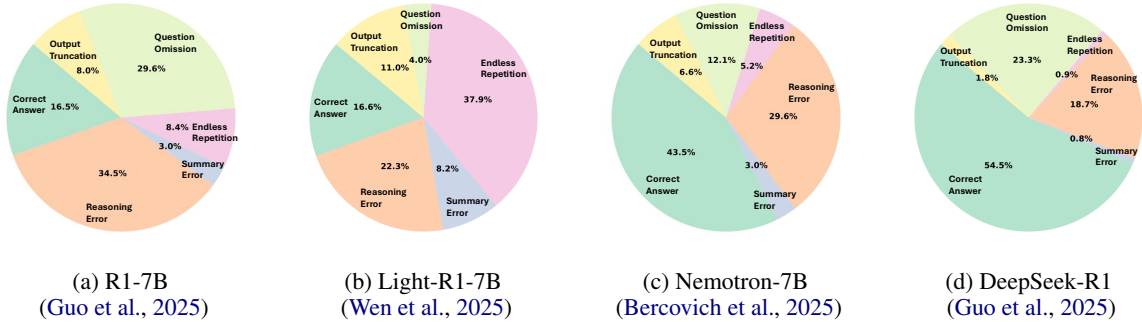


Figure 5: The error type distribution for various LLMs on AIME24 under REST.

tion (OT) becoming more prominent, collectively accounting for 48.90%. This shift underscores the unique challenges of smaller models, whose limited capacity significantly exacerbates performance degradation under the stress imposed by REST. Finally, it is important to note that no Format Violation (FV) errors are found for any of the four models, which is why they are not included in Fig. 5. This observation supports the robustness of our prompt design and answer extraction strategy, confirming that *the observed performance degradation arises primarily from the intrinsic limitations of the LLMs themselves*, rather than artifacts introduced by the evaluation process.

**LLMs that perform better on REST exhibit concise reasoning for earlier questions.** In this paragraph, we investigate why certain models exhibit substantial performance differences under REST, despite achieving similar results on traditional single-question benchmarks. To this end, we analyze responses on AIME24 by manually segmenting each model’s output into question-aligned chunks and measuring the number of reasoning tokens allocated to each. As shown in Fig. 6, high-performing models under REST, such as GPT-OSS-120B (60.28% AIME24 accuracy under REST), **allocate tokens more evenly** across positions and use fewer reasoning tokens per question on average, indicating concise and balanced reasoning. In contrast, lower-performing models like R1-7B (16.53% AIME24 accuracy under REST)

concentrate the majority of their token budget on the first question, leaving insufficient capacity for subsequent ones. This pattern suggests that strong REST performance correlates with the ability to adaptively allocate reasoning effort, employing more concise reasoning on early questions under pressure and preserving resources for later ones. We term this capability as “**adaptive reasoning effort allocation**”, which we believe is a critical factor to achieve high performance under REST.

**Question position affects performance under stress test.** Recent studies (Yu et al., 2024; Liu et al., 2024) have identified positional biases of LLMs over long inputs. In this paragraph, we investigate whether the position of questions within the input can influence accuracy. Fig. 7 presents a heatmap illustrating the accuracy for questions placed at different positions, indicating that *questions appearing earlier consistently achieve higher accuracy*, even under the highest stress levels, while the accuracy for later questions tends to decline as the stress level increases. We attribute this position-dependent degradation to three factors. First, as revealed by our error analysis, Question Omission (QO) is a prevalent failure mode, even for strong LLMs like DeepSeek-R1, leading models to frequently overlook later questions in a prompt. Second, the reasoning trace for earlier questions may act as noise and disrupt subsequent reasoning. Finally, as shown in Fig. 6, early questions consume a disproportionately large por-

Table 4: Comparison between random concatenation and topic-grouped concatenation on MATH500. Topic-grouped concatenation samples all questions in a prompt from the same subject category.

Model	Random Concat.	Topic-grouped Concat.
Qwen3-8B	80.28	80.19
DS-R1-Distill-Qwen-7B	66.75	67.26
OpenR1-Qwen-7B	81.64	81.07
Qwen2.5-Math-7B-Inst	56.59	56.57

tion of the output token budget, leaving insufficient reasoning tokens for later ones.

**Grouping questions by topic does not mitigate contextual interference.** Beyond question position, a natural question is whether the performance degradation is driven by mixing unrelated topics within a single prompt. To investigate this, we compare our default *random concatenation* with a *topic-grouped concatenation* setting on MATH500, where all questions within a prompt are sampled from the same subject category (e.g., Algebra, Number Theory, etc.). As shown in Tab. 4, the performance differences between the two settings are negligible (all  $< 0.6$  points), suggesting that semantic relatedness alone does not substantially reduce contextual interference under REST.

**Attention drift contributes to performance degradation.** To better understand why some models achieve strong single-question performance but degrade sharply under REST, we analyze *attention allocation* when generating the answer for the last question  $Q_N$  in a multi-question prompt. Intuitively, once a model has produced long CoTs for earlier questions  $\{Q_1, \dots, Q_{N-1}\}$ , these self-generated reasoning traces can become strong distractors. If the model continues to attend heavily to them while solving  $Q_N$ , it may fail to fully reset to the current objective.

Quantitatively, let  $A_N = (t_1, \dots, t_i, \dots, t_M)$  denote the generated answer for  $Q_N$ , where  $t_i$  is the  $i$ -th token in  $A_N$ . We measure the attention allocated to three context segments: (1) relevant content  $Q_N \cup A_{N, < t_i}$ , (2) distractor questions  $Q_{< N}$ , and (3) distractor reasoning traces  $A_{< N}$ . For each context segment  $S$ , we define its overall attention weight (averaged across all tokens  $t_i$  in  $A_N$ ) as:

$$\text{Attn}(S) = \frac{1}{LHM} \sum_{l=1}^L \sum_{h=1}^H \sum_{i=1}^M \sum_{k \in (S)} \alpha_{i,k}^{(l,h)}, \quad (1)$$

where  $\alpha_{i,k}^{(l,h)}$  is the attention weight from the  $i$ -th

Table 5: Attention allocation when answering the last question  $Q_N$  under REST. We denote the relative accuracy drop on AIME24 as  $\Delta_{\text{rel}} = \frac{\text{Single-Stress}}{\text{Single}} \times 100\%$ .

Model	$\Delta_{\text{rel}} (\downarrow)$	$\text{Attn}(Q_{< N})$	$\text{Attn}(A_{< N})$	$\text{Attn}(Q_N \cup A_{N, < t_i})$
DS-R1-Distill-Qwen-7B	69.5%	23.2%	25.4%	51.2%
AReaL-boba-RL-7B	65.0%	20.5%	27.2%	52.2%
OpenR1-Qwen-7B	43.6%	19.0%	27.0%	53.9%
Qwen3-8B	40.6%	31.0%	2.1%	63.2%
Qwen2.5-7B-Inst	29.8%	25.3%	4.5%	70.1%

token in  $A_N$  to the  $k$ -th token in the context segment  $S$  at layer  $l$  and head  $h$ , and the weight is averaged over all  $L$  layers,  $H$  heads, and  $M$  generated tokens in the final answer  $A_N$ . Tab. 5 reveals a clear pattern: post-trained models with sharper degradation under REST devote a larger fraction of attention to their previous reasoning traces. For example, AReaL-boba-RL-7B assigns 27.2% attention to  $A_{< N}$  but only 52.2% to the relevant segment  $Q_N \cup A_{N, < t_i}$ , indicating that its generation is strongly distracted by its own earlier reasoning. In contrast, Qwen3-8B, which shows less performance degradation  $\Delta_{\text{rel}}$  under REST, allocates only 2.1% attention to prior reasoning  $A_{< N}$  while maintaining 63.2% on relevant content.

## 5 Related Work

**Compositional Instruction Evaluation** Many real-world applications require LLMs to address multiple queries within a single inference pass. For instance, in some conversation systems, models must respond to multiple queries simultaneously to minimize computational overhead (Lin et al., 2024; Zhu et al., 2025). Similarly, in software development, LLMs are often expected to detect and correct multiple subtle bugs in a code snippet through a single output (Yang et al., 2025b). Several benchmarks have been proposed to evaluate LLMs’ ability to follow compositional instructions. Batch prompting (Cheng et al., 2023; Lin et al., 2024), multi-problem prompting (Wang et al., 2025c), and multi-task inference benchmark (Son et al., 2024) concatenate multiple instructions into a prompt, primarily for improving inference efficiency on simple tasks such as classification and commonsense QA. Benchmark self-evolving (Wang et al., 2025a) complicates the original questions through multi-agent evolving. However, these methods do not involve complex reasoning tasks—a key frontier in contemporary LLM research. In contrast, REST targets reasoning-intensive tasks, where lengthy CoT generation introduces substantial contextual interference that is untested in simpler settings.

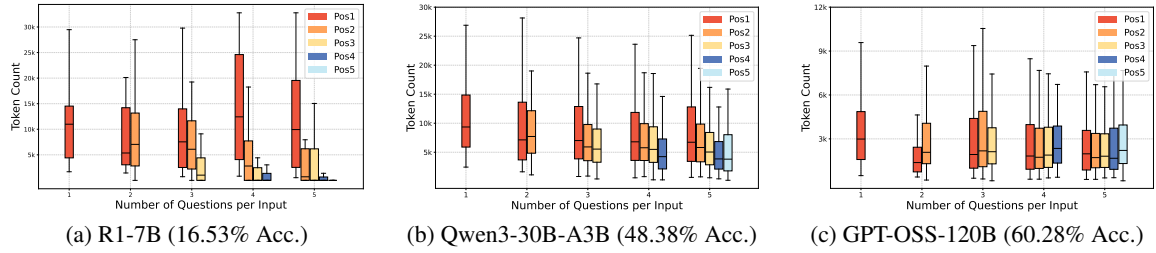


Figure 6: The reasoning token count for questions at different positions on AIME24. The number in parentheses is the accuracy on AIME24 under REST. Results on additional models are reported in Fig. 10 (Appendix).

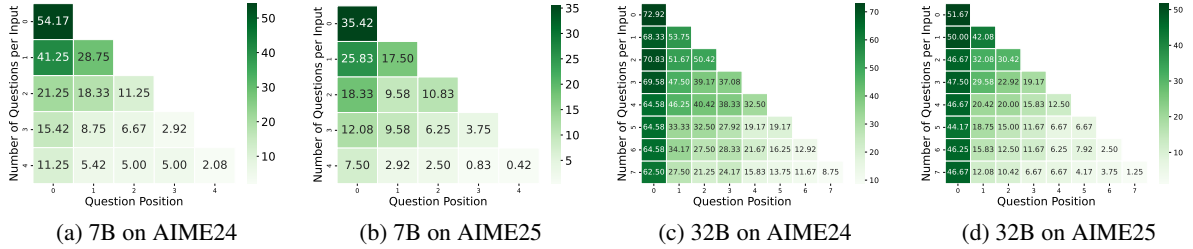


Figure 7: The effect of question position under stress tests. The performance at each position is calculated by averaging the accuracy of questions appearing at the same position across the full benchmark. LRMs generally achieve higher accuracy for earlier questions, while their performance declines for subsequent ones.

**Benchmarks for LRMs** The rigorous assessment of the reasoning ability of LLMs relies heavily on specialized benchmarks. In mathematical reasoning, GSM8K (Cobbe et al., 2021b) and MATH (Hendrycks et al., 2021b) are most commonly used. However, as model capabilities surge, performance on these benchmarks is approaching saturation. Harder benchmarks like AIME (AIMO, 2024; Lin, 2025) and Omni-Math (Gao et al., 2024) have emerged, further pushing the boundaries of mathematical evaluation. Beyond mathematics, reasoning in other domains like code generation (Jain et al., 2024; Quan et al., 2025; Jimenez et al., 2024), cryptographic decryption (Li et al., 2025), and scientific problem-solving (Rein et al., 2024), is also critical. Despite the increasing difficulty of these benchmarks, the rapid improvement of models means that their differentiating power can diminish if evaluated solely in a standard single-problem manner. Our REST, however, can systematically increase the difficulty of existing benchmarks by controlling the stress level.

## 6 Conclusion

In conclusion, we present REST, an empirical and diagnostic framework to stress test LRMs in simultaneous multi-problem solving scenarios. REST substantially sharpens the discriminative power of existing benchmarks, exposing pronounced performance gaps among models that

appear comparable in traditional single-question evaluations. Furthermore, REST uncovers a few reasoning misbehaviors, such as question omission and summary error, which remain obscure in single-question assessments. Critical findings, such as the identification of question position bias, attention drift, and the demonstrated effectiveness of “*Long2Short*” training, offer valuable insights for the future development of more robust LRMs.

## Limitations

While REST provides a powerful and discriminative evaluation framework, several limitations remain. First, REST is primarily empirical and exploratory. It serves as a controlled proxy for increasing reasoning load, and may not fully match real-world tasks. Second, it slightly increases evaluation costs, as multi-problem prompts generally require longer reasoning sequences, leading to increased computational overhead. However, our analysis in Sec. N in Appendix shows that the increase in latency is not significant and is acceptable. Finally, our current evaluation focuses on textual reasoning tasks. Since REST is inherently modality-agnostic, it can be naturally extended to multimodal reasoning scenarios (e.g., processing multiple complex charts or diagrams simultaneously). We leave this extension to future work.

## References

- Pranjal Aggarwal and Sean Welleck. 2025. **L1: Controlling how long a reasoning model thinks with reinforcement learning**. In *Second Conference on Language Modeling*.
- AI-MO. 2024. AIMO Validation AIME Dataset. <https://huggingface.co/datasets/AI-MO/aimo-validation-aime>. Accessed: 2025-03-29.
- Daman Arora and Andrea Zanette. 2025. **Training language models to reason efficiently**. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 114 others. 2025. **Llama-nemotron: Efficient reasoning models**. *Preprint*, arXiv:2505.00949.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. **Do NOT think that much for 2+3=? on the overthinking of long reasoning models**. In *Forty-second International Conference on Machine Learning*.
- Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model apis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. **Training verifiers to solve math word problems**. *Preprint*, arXiv:2110.14168.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.
- Federico Di Maio and Manuel Gozzi. 2025. Degradation of multi-task prompting across six nlp tasks and llm families. *Electronics*, 14(21):4349.
- Bangde Du, Minghao Guo, Songming He, Ziyi Ye, Xi Zhu, Weihang Su, Shuqi Zhu, Yujia Zhou, Yongfeng Zhang, Qingyao Ai, and 1 others. 2025a. Twinvoice: A multi-dimensional benchmark towards digital twins via llm persona simulation. *arXiv preprint arXiv:2510.25536*.
- Bangde Du, Ziyi Ye, Zhijing Wu, Monika A Jankowska, Shuqi Zhu, Qingyao Ai, Yujia Zhou, and Yiqun Liu. 2025b. Simvbg: Simulating individual values by backstory generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13104–13133.
- Marc Eulerich and David A Wood. 2023. A demonstration of how chatgpt can be used in the internal auditing process. *Available at SSRN 4519583*.
- Hugging Face. 2025. **Open r1: A fully open reproduction of deepseek-r1**.
- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omnimath: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021a. **Measuring mathematical problem solving with the math dataset**. *Preprint*, arXiv:2103.03874.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. **Measuring mathematical problem solving with the math dataset**. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Yutao Hou, Yajing Luo, Zhiwen Ruan, Hongru Wang, Weifeng Ge, Yun Chen, and Guanhua Chen. 2024. Compound-qa: A benchmark for evaluating llms on compound questions. *arXiv preprint arXiv:2411.10163*.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xianguyu Zhang, and Heung-Yeung Shum. 2025. Openreasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*.

- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Live-codebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.
- Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. 2025. How many instructions can llms follow at once? In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.
- Yu Li, Qizhi Pei, Mengyuan Sun, Honglin Lin, Chenlin Ming, Xin Gao, Jiang Wu, Conghui He, and Lijun Wu. 2025. Cipherbank: Exploring the boundary of llm reasoning capabilities through cryptography challenges. *arXiv preprint arXiv:2504.19093*.
- Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. 2024. Batchprompt: Accomplish more with less. In *The Twelfth International Conference on Learning Representations*.
- Yen-Ting Lin. 2025. [Aime 2025 dataset](#). Accessed: 2025-03-29.
- Gili Lior, Asaf Yehudai, Ariel Gera, and Liat Ein-Dor. 2025. Wildifeval: Instruction following in the wild. *arXiv preprint arXiv:2503.06573*.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12.
- Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog.
- Zhiyu Mei, Wei Fu, Kaiwei Li, Guangju Wang, Huanchen Zhang, and Yi Wu. 2025. Real: Efficient rlhf training of large language models with parameter reallocation. In *Proceedings of the Eighth Conference on Machine Learning and Systems, MLSys 2025, Santa Clara, CA, USA, May 12-15, 2025*. mlsys.org.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). Preprint, arXiv:2501.19393.
- OpenAI. 2025a. [gpt-oss-120b & gpt-oss-20b model card](#). Preprint, arXiv:2508.10925.
- OpenAI. 2025b. [Openai o3 and o4-mini system card](#).
- Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, Wei Wu, Chenlin Ming, H Vicky Zhao, Conghui He, and Lijun Wu. 2025. Lemma: Learning from errors for mathematical advancement in llms. *arXiv preprint arXiv:2503.17439*.
- Qizhi Pei, Lijun Wu, Zhuoshi Pan, Yu Li, Honglin Lin, Chenlin Ming, Xin Gao, Conghui He, and Rui Yan. 2025. Mathfusion: Enhancing mathematic problem-solving of llm through instruction fusion. *arXiv preprint arXiv:2503.16212*.
- Shanghaoran Quan, Jiayi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, and 1 others. 2025. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings. *arXiv preprint arXiv:2501.01257*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Tushar Semwal, Promod Yenigalla, Gaurav Mathur, and Shivashankar B Nair. 2018. A practitioners’ guide to transfer learning for text classification using convolutional neural networks. In *Proceedings of the 2018 SIAM international conference on data mining*, pages 513–521. SIAM.
- Connor Shorten, Charles Pierse, Thomas Benjamin Smith, Erika Cardenas, Akanksha Sharma, John Trengrove, and Bob van Luijt. 2024. Structuredrag: Json response formatting with large language models. *arXiv preprint arXiv:2408.11061*.
- Guijin Son, SangWon Baek, Sangdae Nam, Ilgyun Jeong, and Seungone Kim. 2024. Multi-task inference: Can large language models follow multiple instructions at once? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5606–5627.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question

- answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North*, page 4149. Association for Computational Linguistics.
- Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025a. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.
- Kimi Team. 2025a. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*.
- Kimi Team, Angang Du, Bofei Gao, BOWEI XING, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025b. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- Ling Team. 2025b. Every attention matters: An efficient hybrid architecture for long-context reasoning. *arXiv preprint arXiv:2510.19338*.
- Open Thoughts Team. 2025c. [Open Thoughts](#).
- Qwen Team. 2025d. [Qwq-32b: Embracing the power of reinforcement learning](#).
- Dietrich Trautmann, Alina Petrova, and Frank Schilder. 2022. Legal prompt engineering for multilingual legal judgement prediction. *arXiv preprint arXiv:2212.02199*.
- Siyuan Wang, Zhuohan Long, Zhihao Fan, Xuan-Jing Huang, and Zhongyu Wei. 2025a. Benchmark self-evolving: A multi-agent framework for dynamic llm evaluation. In *Proceedings of the 31st international conference on computational linguistics*, pages 3310–3328.
- Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025b. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.
- Zhengxiang Wang, Jordan Kodner, and Owen Rambow. 2025c. [Evaluating LLMs with multiple problems at once](#). In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM<sup>2</sup>)*, pages 178–199, Vienna, Austria and virtual meeting. Association for Computational Linguistics.
- Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. 2025. [Light-rl: Curriculum SFT, DPO and RL for long COT from scratch and beyond](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 318–327, Vienna, Austria. Association for Computational Linguistics.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024b. [Qwen2.5-math technical report: Toward mathematical expert model via self-improvement](#). *Preprint*, arXiv:2409.12122.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024c. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.
- Zhiyu Yang, Shuo Wang, Yukun Yan, and Yang Deng. 2025b. [Why stop at one error? benchmarking LLMs as data science code debuggers for multi-hop and multi-bug errors](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17377–17392, Suzhou, China. Association for Computational Linguistics.
- Yixin Ye, Yang Xiao, Tiantian Mi, and Pengfei Liu. 2025. Aime-preview: A rigorous and immediate evaluation framework for advanced mathematical reasoning. <https://github.com/GAIR-NLP/AIME-Preview>. GitHub repository.
- Yijiong Yu, Huiqiang Jiang, Xufang Luo, Qianhui Wu, Chin-Yew Lin, Dongsheng Li, Yuqing Yang, Yongfeng Huang, and Lili Qiu. 2024. Mitigate position bias in large language models via scaling a single dimension. In *First Workshop on Long-Context Foundation Models@ ICML 2024*.
- Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. [Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild](#). *Preprint*, arXiv:2503.18892.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. *Marco-o1: Towards open reasoning models for open-ended solutions*. Preprint, arXiv:2411.14405.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, and 1 others. 2024. *Lmsys-chat-1m: A large-scale real-world llm conversation dataset*. In *The Twelfth International Conference on Learning Representations*.

Xiliang Zhu, Shi Zong, and David Rossouw. 2025. *How accurate are llms at multi-question answering on conversational transcripts?* arXiv preprint arXiv:2509.21732.

## A Further Explanation of the Motivation Behind REST

REST evaluates models by concatenating multiple independent reasoning problems. This design is motivated by the need to simulate the *contextual interference* that arises in real deployments: real-world user requests and professional workflows rarely interact with models through atomic prompts, but instead demand that multiple distinct requirements be tracked and satisfied within a single turn.

First, empirical analyses of large-scale user interaction logs demonstrate that human intent is rarely atomic. Studies such as WildIfEval (Lior et al., 2025) and LMSYS-Chat (Zheng et al., 2024) reveal that real-world users frequently formulate “compound instructions” that impose multiple distinct constraints or sub-goals within a single turn. Several works (Lin et al., 2024; Hou et al., 2024; Di Maio and Gozzi, 2025) explicitly construct compound/batched prompts to mimic this real-world regime. To successfully follow such instructions, the model must maintain attention across disjoint task requirements (e.g., formatting constraints, style guidelines, and content exclusions) simultaneously. REST simulates this cognitive burden by concatenating problems, forcing the model to solve multiple questions within its working memory without succumbing to attention dilution.

Second, this capability is a prerequisite for high-stakes professional domains, particularly in Software Engineering and Legal Compliance. In automated programming, agents are expected to identify multiple distinct errors—ranging from syntax bugs to logic flaws—within a single code review pass, rather than initiating a separate inference cycle for every potential issue (Yang et al., 2025b). Similarly, legal and compliance workflows rely

on “checklist reasoning,” where models must audit documents against dozens of independent criteria (e.g., liability caps, termination clauses, jurisdiction rules) simultaneously (Trautmann et al., 2022; Eulerich and Wood, 2023). REST abstracts these challenges into a standardized format, serving as a proxy for the model’s ability to resist context interference in high-load professional tasks.

Finally, concatenating multiple questions into a single prompt is adopted by some industrial pioneers. For instance, Zhu et al. (2025) study multi-question answering for conversational transcript analysis to increase throughput. Jaroslawicz et al. (2025) likewise highlight that production systems often require models to remain reliable when responding to tens or even hundreds of instructions in one context. Their findings suggest that current LLMs still struggle with multi-instruction following. By adopting the same concatenation paradigm, REST aligns evaluation with practical efficiency constraints in real-world deployment.

Consequently, REST is well-motivated and can be regarded as a proxy for real-world scenarios. Concatenating reasoning problems simulates the contextual interference, and the resulting performance measures how reliably a model can sustain robustness under multiple concurrent objectives in a single context.

## B Differences from Prior Work

Concatenating multiple instructions or problems within a single prompt has been explored in prior work (Cheng et al., 2023; Wang et al., 2025c), often motivated by *inference efficiency* and evaluated only on relatively simple tasks with short, clean outputs (e.g., text classification (Semwal et al., 2018) or commonsense QA (Talmor et al., 2019)). REST adopts the same surface operation (question concatenation) but targets a different goal: mitigating benchmark saturation by systematically increasing *reasoning load* so that near-ceiling single-question benchmarks become discriminative again.

In complex reasoning tasks, lengthy Chain-of-Thought can inject substantial contextual interference. REST imposes a much higher difficulty level by focusing on such tasks, testing the model’s reasoning consistency and robustness in more realistic scenarios. Consequently, our findings challenge the prevalent assumption that “LLMs are multi-problem solvers” (Cheng et al., 2023; Wang et al., 2025c), demonstrating that even SOTA LRMs

exhibit significant degradation under this high-interference setting.

REST also serves as a diagnostic probe beyond prior works, uncovering unique failure modes and training effects. It reveals failure modes that are hidden in single-question evaluation, such as “Question Omission” and “Endless Repetition,” and exposes training-dependent robustness effects—for example, “Long2Short” training can improve multi-problem robustness, whereas standard RL post-training may degrade it.

### C Qualitative Evidence of Contextual Interference

Beyond attention drift, we also observe a qualitative failure mode where the reasoning logic of preceding questions *leaks* into subsequent ones. Concretely, models may fail to reset their working context and incorrectly introduce constraints or concepts that are only relevant to earlier questions, which can lead to hallucinated dependencies and incorrect solutions. Fig. 8 provides an illustrative example: after solving  $Q_3$  with a modular arithmetic argument (modulo 5), the model incorrectly carries over the “modulo 5” notion when answering  $Q_4$ , despite  $Q_4$  being a conditional probability problem without any modular structure.

### D Impact of Prompt Template

In this section, we investigate whether improved prompt design can mitigate performance degradation in multi-question evaluation. Inspired by in-context learning, we enhance the instruction prompt with a few-shot example of the target format: we randomly sample three problems from the MATH training set, generate answers (including reasoning content) using DeepSeek-R1, and concatenate them into a single demonstration of the form “Q1 Q2 Q3 A1 A2 A3”.

As shown in Tab. 18, this few-shot strategy yields modest gains on medium-difficulty benchmarks (i.e., MATH500 and AMC23), but a substantial performance gap relative to the single-question setting persists. This result strongly suggests that the degradation under REST is not primarily due to instruction following or prompt sensitivity, but rather reflects an intrinsic limitation in the model’s reasoning capacity under heightened cognitive load.

### E Impact of Temperature and Top-p

We further examine whether stress sensitivity is an artifact of sampling parameters. On MATH500, we vary temperature  $T$  and top- $p$  under three configurations ( $T=0.1, p=0.5$ ;  $T=0.6, p=0.9$ ;  $T=1.0, p=1.0$ ). As shown in Tab. 6, while absolute accuracies vary across settings, the degradation magnitude (Decrease) remains of similar scale across configurations (e.g., DS-R1-Distill-Qwen-7B drops by 26.25–31.07 points). Together with the prompt template analysis (Tab. 18), this suggests that REST sensitivity is not primarily driven by prompt or sampling choices but a limitation in the model’s intrinsic ability.

### F Question Position Bias beyond AIME

We further examine whether the question position bias in Fig. 7 also generalizes beyond AIME. We extend the analysis to MATH500 and observe a consistent accuracy decline for later questions under REST. For example, DS-R1-Distill-Qwen-7B drops from 56.2% to 39.2%. Fig. 9 visualizes this phenomenon for representative 7B and 32B models.

### G Comparison between Rule-based and LLM-based Answer Extraction

In this section, we compare the performance of several representative LRMs under REST when using rule-based versus LLM-based answer extraction. For rule-based extraction, we search for specific target pattern sequentially using code modified from Qwen-Math evaluation tool<sup>†</sup>. For the LLM-based approach, we employ *gemma-3-27b-it* (Team et al., 2025a) with the prompt shown in Figure 14. The results in Table 7 indicate that LLM-based answer extraction can lead to a modest performance improvement. However, the LLM-based approach significantly increases evaluation costs and may introduce additional instability, making it less practical for large-scale assessments. To further assess the reliability of rule-based extraction, we measure the consistency between the two extraction methods. As shown in Tab. 8, the agreement rates are consistently high ( $> 0.90$ ) and Cohen’s Kappa scores are strong across benchmarks, indicating that the efficient rule-based method serves as a reliable proxy and does not materially change the stress-level conclusions. Therefore, we adopt rule-

<sup>†</sup><https://github.com/QwenLM/Qwen2.5-Math>

**Example C.1: Example of Reasoning Interference (Logic Leakage)**

**Input (excerpt):**

**Q3:** Alice and Bob play the following game. A stack of  $n$  tokens lies before them. The players take turns with Alice going first. On each turn, the player removes 1 token or 4 tokens from the stack. The player who removes the last token wins. Find the number of positive integers  $n \leq 2024$  such that there is a strategy that guarantees that Bob wins, regardless of Alice’s moves.

**Q4:** Jen enters a lottery by picking 4 distinct numbers from  $S = \{1, 2, 3, \dots, 10\}$ . 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 the above questions one by one. Remember to put your final answer within `\boxed{ }`.

**Output (excerpt):**

... (Reasoning for previous questions) ...

... (Reasoning for Q3) ... Thus, for  $n$  from 1 to 2024, the number of losing positions is the number of  $n \equiv 0$  or  $2 \pmod{5}$ .

... (Reasoning for Q4) ... Now, we need to determine the number of outcomes where she “wins a prize.” **However, as we analyzed in the previous problem (Q3), determining a “winning” state requires checking the properties modulo 5.**

... (Reasoning for Q4) ...

Figure 8: A qualitative case of reasoning interference using DS-R1-Distill-Qwen-7B: the modular arithmetic logic used for  $Q_3$  incorrectly leaks into the reasoning for  $Q_4$ , introducing an irrelevant “modulo 5” dependency.

Table 6: Impact of sampling parameters on MATH500. We vary temperature  $T$  and top- $p$  under three configurations, and report single-question accuracy, multi-question accuracy (Stress), and their difference (Decrease).

Model	Single			Stress			Decrease		
	$T=0.1, p=0.5$	$T=0.6, p=0.9$	$T=1.0, p=1.0$	$T=0.1, p=0.5$	$T=0.6, p=0.9$	$T=1.0, p=1.0$	$T=0.1, p=0.5$	$T=0.6, p=0.9$	$T=1.0, p=1.0$
DS-R1-Distill-Qwen-7B	88.80	93.00	90.20	60.14	66.75	59.13	28.66	26.25	31.07
Light-R1-Qwen-7B-DS	92.60	93.20	91.20	62.99	61.73	60.16	29.61	31.47	31.05
OpenR1-Qwen-7B	89.20	92.20	92.00	74.04	81.64	73.74	15.17	10.56	18.26

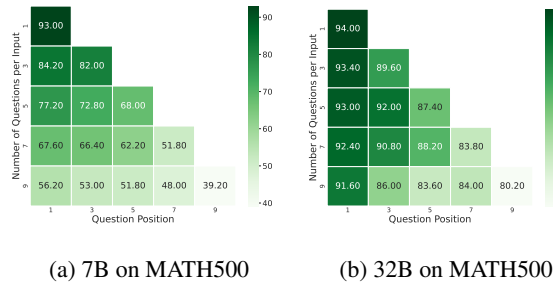


Figure 9: Question position bias on MATH500, accuracy tends to be higher for earlier questions and declines for later ones under multi-question stress.

Table 7: Comparison of rule-based answer extraction and LLM-based answer extraction.

Extraction Method	GSM8K	MATH500	AMC23	AIME24	AIME25
<i>DS-R1-Distill-Qwen-7B</i>					
Rule-Based	89.06	66.75	36.06	16.53	11.37
LLM-Based	90.00	70.59	41.05	18.32	14.16
<i>DS-R1-Distill-Qwen-32B</i>					
Rule-Based	95.50	88.97	86.24	52.51	33.83
LLM-Based	95.83	92.18	87.48	53.63	34.39

based answer extraction as the default approach in our experiments.

Table 8: Agreement between rule-based and LLM-based answer extraction under REST. We report exact-match agreement rate and Cohen’s Kappa.

	GSM8K	MATH500	AMC23	AIME24	AIME25
Agreement	0.945	0.918	0.900	0.916	0.912
Cohen’s Kappa	0.708	0.809	0.789	0.711	0.606

## H Outcome Efficiency Analysis on Overthinking

To quantitatively substantiate the “overthinking” hypothesis, we adopt the outcome efficiency metric  $\xi_O$  (Chen et al., 2025), which measures the fraction of tokens needed to reach the *first* correct answer relative to the total generated tokens. For a dataset of  $N$  instances:

$$\xi_O = \frac{1}{N} \sum_{i=1}^N \sigma_i \frac{\hat{T}_i}{T_i}, \quad (2)$$

where  $T_i$  is the total number of generated tokens for instance  $i$ ,  $\hat{T}_i$  is the number of tokens consumed up to the first correct answer, and  $\sigma_i \in \{0, 1\}$  indicates whether the final prediction is correct. Lower  $\xi_O$  implies more redundant reasoning that does not contribute to correctness. We analyze AIME24 under REST at stress level  $S=5$ , and additionally track the median number of tokens

Table 9: Efficiency analysis on AIME24 at stress level  $S=5$ . “Token Count” reports the median number of generated tokens allocated to the first question  $Q_1$  within a multi-question prompt. “Outcome Efficiency” is the outcome efficiency metric  $\xi_O$  (Chen et al., 2025).

	DS-R1-Distill-Qwen-7B	Qwen3-30B-A3B	GPT-OSS-120B
AIME24 Acc.	16.53%	48.38%	60.28%
Token Count (Median, $Q_1$ )	9987	6818	2108
Outcome Efficiency ( $\xi_O$ )	14.16%	28.37%	63.22%

Table 10: Evaluation results on ARC-C and BBH. The “Stress” column reports the average performance across four distinct stress levels, as described in Sec. 3.1. Models that perform best under “Stress” are highlighted in gray.

Model \ Bench	ARC-C		BBH	
	Single	Stress	Single	Stress
<i>1.5B LRMs</i>				
DeepSeek-R1-Distill-Qwen-1.5B	62.37	50.70	59.60	18.58
DeepScaleR-1.5B	66.44	48.39	74.80	26.94
L1-Qwen-1.5B-Max	63.39	53.59	73.60	40.75
Qwen2.5-1.5B-Inst	78.64	68.64	32.80	3.74
Qwen3-1.7B	<b>87.46</b>	<b>79.20</b>	<b>86.00</b>	<b>62.64</b>
<i>7 ~ 8B LRMs</i>				
DeepSeek-R1-Distill-Qwen-7B	83.73	77.83	88.40	57.40
DeepSeek-R1-Distill-LLaMA-8B	86.78	75.04	94.40	60.32
Efficient-R1-7B ( $\alpha = 0.2$ )	83.39	77.24	89.20	57.24
Nemotron-Nano-8B	78.64	55.79	50.80	43.85
AReal-boba-RL-7B	88.14	72.57	92.40	49.49
Light-R1-7B-DS	81.69	76.34	81.60	51.07
OpenR1-Qwen-7B	79.66	69.90	85.60	54.25
OpenThinker2-7B	91.53	87.85	95.20	82.55
Qwen2.5-7B-Inst	91.19	88.93	60.40	45.65
Qwen3-8B	<b>94.58</b>	<b>91.25</b>	<b>98.00</b>	<b>89.41</b>
<i>32B LRMs</i>				
DeepSeek-R1-Distill-Qwen-32B	94.58	94.66	97.20	96.11
Qwen-QwQ-32B	<b>94.92</b>	93.50	96.80	<b>96.99</b>
Qwen2.5-32B-Inst	94.58	<b>94.92</b>	74.80	65.66
Qwen3-32B	94.58	94.13	<b>99.20</b>	88.34
<i>Proprietary LRMs</i>				
DeepSeek-R1	<b>97.29</b>	94.79	<b>99.60</b>	<b>97.36</b>
O3-mini	95.93	<b>95.14</b>	95.20	89.59
O4-mini	95.93	94.53	97.20	94.11
Gemini-2.5-Flash-Thinking	96.27	94.44	97.60	86.72

allocated to the first question  $Q_1$  within the concatenated prompt. As shown in Tab. 9, models with lower accuracy under REST (DS-R1-Distill-Qwen-7B) exhibit substantially lower outcome efficiency (14.16%) along with excessive token usage (9987 tokens) on early questions, while models with higher accuracy (Qwen3-30B-A3B and GPT-OSS-120B) exhibit more efficient reasoning (28.37% and 63.22%). This result is consistent with the “overthinking trap” hypothesis.

## I Evaluation Results on Other Tasks

To further demonstrate broader applicability, we conduct additional experiments on (1) AI2 Rea-

soning Challenge – Challenge (ARC-C) (Clark et al., 2018), including questions that require reasoning and **commonsense knowledge** to answer. (2) Big Bench Hard (BBH) (Suzgun et al., 2023), which evaluates logical reasoning. Due to the computation resource constraints, we select the “logical\_deduction\_five\_objects” subset to represent BBH, as it specifically focuses on **logical reasoning**. The results, presented in Tab. 10, strongly corroborate our findings in the main text:

(1) **Substantial performance degradation**: All models exhibit performance degradation under the multi-problem stress of REST. While some top-tier models like DeepSeek-R1 largely retain their single-question accuracy, smaller models such as DS-R1-Distill-Qwen-7B suffer severe performance drops, especially on BBH, where its accuracy declines from 88.40% to 57.40%.

(2) **Enhanced discriminative power**: REST significantly amplifies performance differences between models, thereby enhancing the discriminative power of existing benchmarks. For example, on BBH, the accuracy of AReal-boba-RL-7B plummets from 92.40% to just 49.49% under REST, a decline of 42.91%. In contrast, the larger model DS-R1-Distill-Qwen-32B retains more of its single-question performance, with its accuracy under REST only dropping from 97.20% to 96.11%, a reduction of merely 1.09%

(3) **The benefit of “Long2Short” training**: Models trained with the “Long2Short” technique consistently demonstrate more robust performance under stress compared to their standard-trained counterparts, confirming our initial findings. For instance, the accuracy of L1-Qwen-1.5B-Max on BBH is 40.75% under REST, which is much higher than the 18.58% and 26.94% achieved by R1-1.5B and DeepScaleR-1.5B, respectively.

## J Evaluation Results on Additional Models

Table 12 and Table 11 present the evaluation results on additional models. These results again validate our core findings: REST induces significant performance degradation across diverse model sizes and architectures, and the performance gap between models is substantially amplified under multi-question stress, with smaller models exhibiting notably sharper declines. Additionally, to investigate whether emerging attention mechanisms can mitigate the contextual interference observed under

Table 11: Evaluation results on GPQA-Diamond and LiveCodeBench (v5) for additional models. The “Stress” column reports the average performance across four distinct stress levels, as described in Sec. 3.1. Models that perform best under “Stress” are highlighted in gray.

Model \ Bench	GPQA-Diamond		LiveCodeBench (v5)	
	Single	Stress	Single	Stress
<i>1.5B LRMs</i>				
DeepSeek-R1-Distill-Qwen-1.5B	<b>37.37</b>	22.11	15.05	0.48
DeepScaleR-1.5B	31.82	27.90	21.15	1.83
L1-Qwen-1.5B-Exact	33.84	31.01	18.28	2.70
L1-Qwen-1.5B-Max	36.87	32.03	19.35	2.45
Qwen2.5-1.5B-Inst	26.26	21.52	1.79	0.37
Qwen3-1.7B	<b>37.37</b>	<b>32.21</b>	<b>30.82</b>	<b>6.65</b>
<i>7 ~ 8B LRMs</i>				
DeepSeek-R1-Distill-Qwen-7B	51.01	31.67	37.63	2.89
DeepSeek-R1-Distill-LLaMA-8B	50.00	33.99	39.43	11.25
Efficient-R1-7B ( $\alpha = 0.2$ )	47.97	34.37	38.71	2.50
Nemotron-Nano-8B	51.01	34.06	50.90	8.56
AReal-boba-RL-7B	48.98	29.13	37.99	4.73
Light-R1-7B-DS	41.91	30.32	39.07	2.85
OpenR1-Qwen-7B	38.38	36.04	4.66	1.27
OpenThinker2-7B	49.49	40.60	39.43	14.51
SimpleRL-Zoo-Qwen-7B	33.84	35.49	5.73	0.16
Open-Reasoner-Zero-7B	37.37	34.25	16.13	0.64
Marco-O1-7B	30.81	28.32	9.32	7.24
Qwen2.5-7B-Inst	35.86	35.15	13.98	10.19
Qwen3-8B	<b>61.11</b>	<b>58.60</b>	<b>56.99</b>	<b>31.13</b>
<i>32B LRMs</i>				
DeepSeek-R1-Distill-Qwen-32B	60.10	53.73	55.56	26.71
Qwen-QwQ-32B	63.64	<b>60.03</b>	62.37	32.16
AReal-boba-SFT-32B	63.13	50.59	60.93	31.74
Light-R1-32B-DS	<b>65.66</b>	50.11	60.93	28.22
S1.1-32B	61.62	54.54	25.45	24.46
OpenThinker2-32B	62.12	57.79	56.27	<b>37.53</b>
SimpleRL-Zoo-Qwen-32B	46.46	46.21	26.52	23.95
Open-Reasoner-Zero-32B	60.10	49.57	35.13	13.01
OpenThinker2-32B	62.12	57.79	56.27	37.53
Qwen2.5-32B-Inst	42.93	40.04	26.88	24.00
Qwen3-32B	63.33	48.13	<b>62.72</b>	30.98
<i>MoE LRMs</i>				
GPT-OSS-20B	53.54	59.52	43.37	29.83
GPT-OSS-120B	60.10	<b>65.93</b>	63.08	<b>49.16</b>
Qwen3-30B-A3B	64.65	50.74	61.65	38.48
DeepSeek-R1	<b>70.20</b>	64.63	<b>63.44</b>	40.83
<i>Proprietary LRMs</i>				
O3-mini	71.21	67.39	60.21	48.36
O4-mini	76.26	<b>73.11</b>	<b>70.61</b>	<b>63.07</b>
Gemini-2.5-Flash-Thinking	<b>78.79</b>	68.00	61.65	48.34

REST, we evaluate models employing two recent architectural innovations: *Dual Chunk Attention* (Qwen3-32B (Yang et al., 2025a)) and *Hybrid Attention* that combines linear and standard attention (Ring-Flash-Linear-2.0 (Team, 2025b) and Kimi-Linear (Team, 2025a)).

As shown in Tab. 13, these architectures do not resolve the performance degradation under REST. While Ring-Flash-Linear-2.0 achieves a moderate drop of 8.89% on MATH500, it suffers a severe 51.39% decline on AIME25. Kimi-Linear exhibits even sharper degradation across both benchmarks (39.81% and 23.71%). These results suggest that

the contextual interference exposed by REST is a fundamental challenge for LRMs that cannot be addressed by attention mechanism design alone.

## K Additional Analysis on Per Question Token Allocation

We analyze how models distribute reasoning tokens across question positions under multi-question stress. As shown in Fig. 10, strong models under REST, such as DeepSeek-R1, tend to *use fewer reasoning tokens on the first question* when the stress level exceeds 1. In contrast, models with lower REST performance, such as Light-R1-7B-DS, often *overthink the first question*, consuming a substantial portion of their reasoning tokens and leaving insufficient space for subsequent questions. These findings on additional models again reinforce our conclusion in the main text: “**adaptive reasoning effort allocation**”—the ability to dynamically distribute reasoning resources across questions—is critical for building strong, multi-question capable LRMs.

## L Detailed Evaluation Results

We present the detailed accuracy statistics for each stress level of various LRMs in Table 19, Table 20 and Table 21. The stress degree  $\{I, II, III, IV, V\}$  corresponds to different stress levels in each dataset. For GSM8K, ARC-C and BBH:  $\{1, 3, 6, 9, 12\}$ ; for MATH500 and AMC23:  $\{1, 3, 5, 7, 9\}$ ; for AIME24, AIME25, GPQA and LiveCodeBench:  $\{1, 2, 3, 4, 5\}$ . We use the official test splits for all benchmarks except ARC-C, for which we follow OpenCompass’s standard practice and use the development split (see opencompass<sup>‡</sup>).

## M The Impact of Maximum Output Length

**Extending the maximum output length can not provide substantial improvement.** As our error analysis identified output truncation as a significant error type, it is questionable whether extending the maximum output length could improve performance under stress tests. To evaluate this, we extended the maximum output length from 32K to 128K tokens for top-tier 7B~32B reasoning models. As shown in Tab. 14, this extension has a negligible impact on stress test accuracy.

<sup>‡</sup><https://github.com/open-compass/opencompass>

Table 12: Evaluation results on various math benchmarks for additional models. The ‘‘Stress’’ column reports the average performance across four stress levels, as described in Sec. 3.1. Models that perform best under ‘‘Stress’’ are highlighted in gray. † denotes models that are only partially evaluated (on AIME24 and AIME25) due to API cost.

Bench Model	GSM8K		MATH500		AMC23		AIME24		AIME25		AVG.	
	Single	Stress	Single	Stress	Single	Stress	Single	Stress	Single	Stress	Single	Stress
<i>1.5B LRMs</i>												
DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025)	84.62	70.21	83.40	42.47	62.50	13.98	29.17	4.97	25.00	5.91	56.94	27.51
DeepScaleR-1.5B (Luo et al., 2025)	84.84	66.58	87.60	59.77	76.25	32.05	38.75	12.82	31.25	14.23	63.74	37.09
L1-Qwen-1.5B-Exact (Aggarwal and Welleck, 2025)	84.87	79.01	84.00	72.07	71.25	47.37	21.25	12.62	18.33	12.96	55.94	44.81
L1-Qwen-1.5B-Max (Aggarwal and Welleck, 2025)	84.17	78.29	83.40	<b>73.23</b>	77.50	<b>48.37</b>	20.00	15.13	22.92	14.95	57.60	<b>45.99</b>
Qwen2.5-Math-1.5B-Inst (Yang et al., 2024c)	85.37	67.49	73.00	53.94	57.50	22.22	10.83	6.17	10.83	2.83	47.51	30.53
Qwen3-1.7B (Yang et al., 2025a)	<b>90.22</b>	<b>85.26</b>	<b>90.20</b>	62.13	<b>85.00</b>	40.43	<b>46.67</b>	<b>23.61</b>	<b>35.83</b>	<b>17.13</b>	<b>69.58</b>	45.71
<i>7 ~ 8B LRMs</i>												
DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025)	89.49	89.06	93.00	66.75	87.50	36.06	54.17	16.53	35.42	11.37	71.92	43.95
DeepSeek-R1-Distill-LLaMA-8B (Guo et al., 2025)	90.45	85.18	89.80	81.34	87.50	70.75	50.42	31.23	28.33	22.66	69.30	58.23
Efficient-R1-7B ( $\alpha = 0.1$ ) (Arora and Zanette, 2025)	88.63	84.76	90.00	74.99	87.50	44.25	54.58	21.45	35.42	15.75	71.23	48.24
Efficient-R1-7B ( $\alpha = 0.2$ ) (Arora and Zanette, 2025)	87.95	80.38	88.20	76.41	85.00	48.05	50.42	22.12	33.75	17.25	69.06	48.84
Nemotron-Nano-8B (Bercovich et al., 2025)	91.36	70.52	94.40	<b>86.04</b>	90.00	<b>76.24</b>	63.33	43.55	50.00	<b>32.28</b>	77.82	<b>61.71</b>
AReal-boba-RL-7B (Mei et al., 2025)	91.66	77.80	95.00	60.77	91.25	32.94	61.25	21.43	45.83	12.33	77.00	41.05
Light-R1-7B-DS (Wen et al., 2025)	88.05	82.69	93.20	61.73	90.00	34.91	55.83	16.63	45.83	12.96	74.58	41.78
OpenR1-Qwen-7B (Face, 2025)	<b>95.60</b>	90.22	92.20	81.64	83.75	54.11	47.50	26.77	32.92	21.19	70.39	54.79
OpenThinker2-7B (Team, 2025c)	94.39	91.99	93.80	83.30	85.00	63.23	54.58	34.50	41.67	23.66	73.89	59.33
SimpleRL-Zoo-Qwen-7B (Zeng et al., 2025)	90.52	84.01	77.80	62.41	68.50	16.16	26.67	7.55	10.00	6.47	54.70	35.32
Open-Reasoner-Zero-7B (Hu et al., 2025)	92.87	65.14	83.00	32.51	60.00	31.23	17.92	6.13	16.25	3.89	54.01	27.78
Marco-O1-7B (Zhao et al., 2024)	89.08	79.56	72.40	48.19	47.50	17.23	10.00	4.35	10.83	3.64	45.96	30.59
MathFusion-Qwen-7B (Pei et al., 2025)	89.46	83.78	74.00	68.15	52.50	36.24	9.58	7.89	5.83	2.35	46.27	39.68
Eurus-2-7B-PRIME (Cui et al., 2025)	92.72	88.01	81.40	64.69	62.50	38.58	20.83	10.84	14.58	4.49	54.41	41.32
Qwen2.5-Math-7B-Inst (Yang et al., 2024b)	95.53	78.53	83.60	56.59	60.00	28.46	14.17	6.40	11.67	5.33	52.99	35.06
Qwen2.5-7B-Inst (Yang et al., 2024a)	92.27	85.12	77.60	65.78	42.50	34.46	10.00	7.02	3.75	3.32	45.22	39.14
Qwen3-8B (Yang et al., 2025a)	95.30	<b>93.90</b>	<b>95.40</b>	80.28	<b>92.50</b>	53.59	<b>74.58</b>	<b>44.28</b>	<b>63.33</b>	32.06	<b>84.22</b>	60.82
<i>32B LRMs</i>												
DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025)	95.54	95.50	94.60	88.97	94.75	<b>86.24</b>	72.92	52.51	51.67	33.83	81.90	71.41
Qwen-QwQ-32B (Team, 2025d)	95.83	<b>95.78</b>	96.20	<b>92.49</b>	95.00	82.89	78.75	54.79	69.58	41.53	87.07	<b>73.49</b>
AReal-boba-SFT-32B (Mei et al., 2025)	95.01	94.75	95.00	88.92	<b>97.50</b>	78.96	77.50	45.79	60.00	33.55	85.00	68.39
Light-R1-32B-DS (Wen et al., 2025)	95.83	94.79	95.60	83.66	96.25	68.80	77.50	41.26	60.00	33.80	85.04	64.46
S1.1-32B (Muennighoff et al., 2025)	89.84	61.10	90.40	53.85	90.00	32.26	55.83	24.42	45.42	19.13	74.30	38.15
OpenThinker2-32B (Team, 2025c)	<b>96.44</b>	95.17	96.20	90.10	95.00	81.00	68.33	53.01	52.50	38.20	81.69	71.49
SimpleRL-Zoo-Qwen-32B (Zeng et al., 2025)	96.06	93.49	83.20	78.90	67.50	57.02	27.20	16.80	16.67	8.87	58.13	51.01
Open-Reasoner-Zero-32B (Hu et al., 2025)	95.83	91.80	92.00	82.90	83.75	70.04	46.67	31.65	36.67	23.63	70.98	60.00
Qwen2.5-32B-Inst (Yang et al., 2024a)	95.53	93.77	82.20	73.39	60.00	49.72	20.00	9.61	16.67	6.73	54.88	46.64
Qwen3-32B (Yang et al., 2025a)	95.91	93.81	<b>96.60</b>	83.64	93.75	80.62	<b>85.00</b>	<b>56.67</b>	<b>71.67</b>	<b>44.53</b>	<b>88.59</b>	71.85
<i>MoE LRMs</i>												
GPT-OSS-20B (OpenAI, 2025a)	88.02	49.31	88.40	46.96	97.50	56.56	66.25	36.20	63.33	29.87	80.70	43.78
GPT-OSS-120B (OpenAI, 2025a)	84.91	47.65	89.60	60.06	93.75	71.93	67.08	<b>60.28</b>	68.33	<b>52.39</b>	80.73	58.46
Qwen3-30B-A3B (Yang et al., 2025a)	96.13	94.91	96.40	80.36	<b>98.75</b>	72.34	79.58	48.38	<b>68.75</b>	28.67	<b>87.92</b>	64.93
DeepSeek-R1 (Guo et al., 2025)	<b>96.20</b>	<b>96.16</b>	<b>97.00</b>	<b>92.09</b>	93.75	<b>81.80</b>	<b>81.66</b>	52.49	<b>68.75</b>	37.17	87.47	<b>71.94</b>
<i>Proprietary LRMs</i>												
O3-mini (OpenAI, 2025b)	95.83	93.85	95.00	86.62	90.00	59.17	79.16	34.07	71.66	20.63	86.25	55.51
O4-mini (OpenAI, 2025b)	93.71	93.07	90.00	82.40	96.25	<b>82.79</b>	73.33	49.69	<b>80.00</b>	<b>41.42</b>	86.66	69.87
O3 (OpenAI, 2025b)†	-	-	-	-	-	-	<b>83.33</b>	<b>76.60</b>	86.67	71.56	-	-
Gemini-2.5-Flash-Thinking (Team et al., 2023)	89.23	91.28	<b>97.20</b>	69.92	<b>97.50</b>	47.63	76.67	26.60	71.67	16.54	86.45	50.39
Gemini-3-Pro (Team et al., 2023)†	-	-	-	-	-	-	96.67	87.90	<b>96.67</b>	<b>75.11</b>	-	-

Table 13: Evaluation of emerging attention architectures under REST. ‘‘Drop’’ denotes the absolute accuracy decrease from Single to Stress.

Bench Model	Architecture	MATH500			AIME25		
		Single	Stress	Drop	Single	Stress	Drop
Qwen3-32B (Yang et al., 2025a)	Dual Chunk Attn	96.60	83.64	12.96	71.67	44.53	27.14
Ring-Flash-Linear-2.0 (Team, 2025b)	Hybrid Attn	96.00	87.11	8.89	72.92	21.53	51.39
Kimi-Linear (Team, 2025a)	Hybrid Attn	71.80	31.99	39.81	28.33	4.63	23.71

## N Computation Overhead Analysis

Our primary goal is to steer the development of more *robust and reliable* reasoning systems for

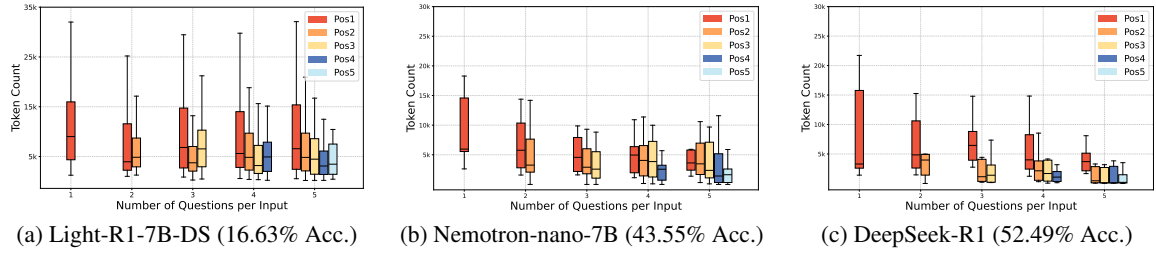


Figure 10: The reasoning token count for questions at different positions on AIME24 for additional models. The number in parentheses indicates the accuracy on AIME24 under REST.

Max Output Token	GSM8K	MATH500	AMC23	AIME24	AIME25
<i>DS-R1-Distill-Qwen-7B</i>					
32K	89.06	66.75	36.06	16.53	11.37
128K	89.37	65.83	36.66	16.17	11.73
<i>DS-R1-Distill-Qwen-32B</i>					
32K	95.50	88.97	86.24	52.51	33.83
128K	95.62	89.71	86.77	52.75	33.87

Table 14: The effect of extending the maximum output length. Extending the maximum output length has almost negligible impact on both 7B and 32B models.

Table 15: Average inference latency (seconds) of DS-R1-Distill-Qwen-7B on AIME24 under different stress levels, profiled on a single NVIDIA H100 GPU.

Stress level	1 (Standard)	2	3	4	5
Latency (s)	694	818	1003	1091	1131

complex, real-world multi-task scenarios, where models must prioritize and solve multiple requests under contextual pressure. At the same time, REST may introduce additional computational cost as LRMs need to generate more reasoning tokens for multiple questions. To quantify the overhead, we profile the average inference latency of DS-R1-Distill-Qwen-7B on AIME24 using a single NVIDIA H100 GPU. As shown in Tab. 15, latency increases with stress levels but remains moderate (stress level 5 is within  $2\times$  the latency of the standard setup), suggesting a practical trade-off for improved evaluation robustness.

## O Prompts

We present the prompt for mathematical tasks, *GPQA Diamond* and *LiveCodeBench* in Figure 11, Figure 12 and Figure 13, respectively.

## P Extended Error Analysis on Other Models and Benchmarks

To broaden the scope of our error analysis, we extend our evaluation to: (1) **Additional models**: the smaller DeepSeek-R1-Distill-Qwen-1.5B and the

### Prompt 1: Prompt for Math Tasks

{Questions}. Answer the above questions one by one. Remember to put your final answer within `\boxed{ }`.

Figure 11: Evaluation prompt for math tasks.

### Prompt 2: Prompt for GPQA Diamond

{Questions}. Answer the above multiple-choice question one by one. Remember to give each answer in the following format: ‘ANSWER: `\boxed{LETTER}`’ (without quotes) where LETTER is one of ABCD.

Figure 12: Evaluation prompt for GPQA Diamond.

### Prompt 3: Prompt for Code Generation

{Questions}. Answer the above questions one by one. Enclose the code for each question within delimiters as follows.  
````python #YOUR CODE HERE ````.  
`### Answer: (use the provided format with backticks)`

Figure 13: Evaluation prompt for code generation tasks.

non-reasoning model Qwen2.5-7B-Instruct; and (2) **Simpler benchmarks**: GSM8K and MATH500. Given the substantial volume of data for error analysis, we employ Qwen3-32B for error cause annotation. We randomly sample 10% of the annotated data for manual human review, achieving 98.25% agreement with human labels, which verifies the reliability of this LLM-based error analysis.

As summarized in Tab. 16, this expansion reveals notable shifts in error distributions. For instance, Qwen2.5-7B-Instruct exhibits substantially fewer length-related errors—such as *Question Omission*, *Output Truncation*, and *Endless Repetition*—but shows a marked increase in *Reasoning Errors*. We

#### Prompt 4: Answer Extraction Prompt

Extract the final answers from the given predictions. Here are some extraction criteria:

1. Don't try to answer the original question. Your task is to extract the final answer from the prediction as it is, even if it is incorrect.
2. Prediction sometimes involves lengthy thinking processes, you don't need to consider these, just extract the final answer.
3. If there is question that has not been answered, don't answer it yourself. You should set the final answer to None (e.g., `\boxed{None}`).
4. Ensure the number of final answers you extract is exactly the same as the number of the given questions.
5. Extract the final answer for each question one by one and enclose each final answer within an `\boxed`. For example, if there are three questions, the output should be

Answer to Q1: `\boxed{answer 1}`

Answer to Q2: `\boxed{answer 2}`

Answer to Q3: `\boxed{answer 3}`.

Here is your task. Simply extract the final answers from the given predictions. Don't apologize or correct yourself if there was a mistake in the predictions; we are just trying to extract the final answer.

<Questions Begin>

{question}

<Questions End>

<Predictions Begin>

{prediction}

<Predictions End>

Extract the final answers from the given predictions.

Figure 14: Prompt for extracting answers from the response.

attribute this to its non-reasoning nature: by using shorter Chain-of-Thought (CoT), it avoids length-related errors but often fails to perform sufficient deliberation.

Furthermore, on simpler benchmarks like GSM8K and MATH500, reasoning misbehaviors such as *Question Omission*, *Summary Error*, and *Endless Repetition* are significantly reduced. This indicates that these degradation patterns are triggered by the increased reasoning load of more challenging questions (e.g., AIME24). Nevertheless, DeepSeek-distilled models still exhibit non-negligible rates of such errors, and their prevalence remains substantially higher than in Qwen2.5-7B-Instruct. This contrast underscores a key trade-off: while extended Chain-of-Thought (CoT) reasoning enhances single problem solving capability, it simultaneously compromises robustness under multi-question stress.

### Q Specific Error Type Reduction via “Long2Short” Training

We conduct experiments to probe the specific failure types mitigated by “Long2Short” training. The results in Tab. 17 confirm that “Long2Short” training reduces actual reasoning failures, not just length-related artifacts. Specifically, we observe a notable decrease in “Question Omission” (QO) and “Reasoning Error” (RE) for models after “Long2Short” training. For example, on MATH500, the Long2Short-trained L1-Qwen-1.5B-Max shows a lower percentage of “Question Omission” and “Reasoning Error” (3.85% and 20.60%, respectively) compared to DS-R1-Distill-Qwen-1.5B (15.10% and 33.85%). A similar pattern is also observed in the 7B models (Efficient-R1-7B ( $\alpha = 0.2$ ) vs. DS-R1-Distill-Qwen-7B). These observations indicate that “Long2Short” training not only prevents output truncation by shortening the thinking process but also effectively mitigates degradation patterns in reasoning, thus improving model robustness under multi-question stress.

Error Type Model	Question Omission	Summary Error	Endless Repetition	Output Truncation	Reasoning Error	Format Violation	Correct Answer
	<i>GSM8K</i>						
DS-R1-Distill-Qwen-1.5B	4.8	3.1	0.7	0.5	20.6	0.1	70.2
DS-R1-Distill-Qwen-7B	2.2	2.0	0.0	0.2	6.5	0.1	89.0
Qwen2.5-7B-Instruct	0.4	0.8	0.3	0.0	13.4	0.0	85.1
<i>MATH500</i>							
DS-R1-Distill-Qwen-1.5B	15.1	5.6	1.3	1.4	33.9	0.2	42.5
DS-R1-Distill-Qwen-7B	10.4	4.7	0.3	0.7	17.0	0.1	66.8
Qwen2.5-7B-Instruct	4.5	0.6	2.7	1.0	25.2	0.2	65.8
<i>AIME24</i>							
DS-R1-Distill-Qwen-1.5B	38.6	5.9	0.9	3.6	45.8	0.2	5.0
DS-R1-Distill-Qwen-7B	29.6	3.0	8.4	8.0	34.5	0.0	16.5
Qwen2.5-7B-Instruct	18.5	4.2	5.8	3.3	61.2	0.0	7.0

Table 16: Error analysis for other models on more benchmarks.

Error Type Model	Question Omission	Reasoning Error	Summary Error	Endless Repetition	Format Violation	Output Truncation	Correct Answer
	<i>1.5B models on MATH500</i>						
DS-R1-Distill-Qwen-1.5B	15.10	33.85	5.65	1.35	0.20	1.38	42.47
L1-Qwen-1.5B-Max	3.85	20.60	2.17	0.05	0.10	0.00	73.23
<i>7B models on MATH500</i>							
DS-R1-Distill-Qwen-7B	10.40	17.00	4.75	0.35	0.05	0.70	66.75
Efficient-R1-7B ( $\alpha = 0.2$ )	5.40	14.20	3.35	0.14	0.10	0.40	76.41
<i>7B models on AMC23</i>							
DS-R1-Distill-Qwen-7B	30.62	25.62	4.37	0.84	0.62	1.87	36.06
Efficient-R1-7B ( $\alpha = 0.2$ )	21.87	24.00	2.81	0.77	1.25	1.25	48.05

Table 17: Error type distribution comparison between standard-trained and Long2Short-trained models.

Table 18: Evaluation results with the few-shot prompt. The ‘‘Few-Shot’’ column reports the average performance across four stress levels using the few-shot prompt described in Sec. D.

Bench Model	GSM8K		MATH500		AMC23		AIME24		AIME25		AVG.	
	Zero-Shot	Few-Shot	Zero-Shot	Few-Shot	Zero-Shot	Few-Shot	Zero-Shot	Few-Shot	Zero-Shot	Few-Shot	Zero-Shot	Few-Shot
<i>7B LRMs</i>												
DS-R1-Distill-Qwen-7B	89.06	88.48	66.75	70.77	36.06	35.88	16.53	14.16	11.37	12.29	43.95	44.32
OpenThinker2-7B	91.99	91.79	83.30	85.72	63.23	71.32	34.50	33.36	23.66	21.81	59.34	60.80
OpenR1-Qwen-7B	90.22	90.03	81.64	81.80	54.11	64.56	26.77	24.13	21.19	19.67	54.79	56.04
Light-R1-7B-DS	82.69	80.60	61.73	66.29	34.91	35.53	16.63	14.97	12.96	11.60	41.78	41.80
Qwen2.5-Math-7B-Inst	78.53	82.65	56.59	61.46	28.46	28.51	6.40	6.19	5.33	4.72	35.06	36.71

## R Error Type Illustrations

In this section, we provide a detailed case study of each error type under stress tests in figs. 15 to 19. Our findings indicate that *Output Truncation* is not the only reason for the performance degradation under REST, revealing some unexpected misbehaviors like *Question Omission* and *Summary Error* that remain hidden in standard evaluation settings.

## S Use of AI Assistance

We used large language models to assist with language polishing and editing of the manuscript. All technical content, analysis, and conclusions were developed and verified by the authors.

**Example S.1: Example of Output Truncation**

**Query:**

**Q1:** Let  $O(0, 0)$ ,  $A(\frac{1}{2}, 0)$ , and  $B(0, \frac{\sqrt{3}}{2})$  be points in the coordinate plane. Let  $\mathcal{F}$  be the family of segments  $\overline{PQ}$  of unit length lying in the first quadrant with  $P$  on the  $x$ -axis and  $Q$  on the  $y$ -axis. There is a unique point  $C$  on  $\overline{AB}$ , distinct from  $A$  and  $B$ , that does not belong to any segment from  $\mathcal{F}$  other than  $\overline{AB}$ . Then  $OC^2 = \frac{p}{q}$ , where  $p$  and  $q$  are relatively prime positive integers. Find  $p + q$ .

**Q2:** Find the number of triples of nonnegative integers  $(a, b, c)$  satisfying  $a + b + c = 300$  and  $a^2b + a^2c + b^2a + b^2c + c^2a + c^2b = 6,000,000$ .

Answer the above questions one by one. Remember to put your final answer within `\boxed{}`.

**Response:**

Okay, so I have these two problems to solve. Let me start with the first one.

**Problem Q1:**

We have points  $O(0,0)$ ,  $A(1/2, 0)$ , and  $B(0, \sqrt{3}/2)$ . The family  $F$  consists of segments  $PQ$  of unit length in the first quadrant, with  $P$  on the  $x$ -axis and  $Q$  on the  $y$ -axis. We need to find a unique point  $C$  on  $AB$  (other than  $A$  and  $B$ ) that doesn't lie on any other segment from  $F$  except  $AB$ . Then compute  $OC^2$  and express it as a reduced fraction  $p/q$ , and find  $p+q$ . Alright, let's break this down. First, let's visualize the points.  $O$  is the origin,  $A$  is  $(1/2, 0)$ , which is halfway between  $(0,0)$  and  $(1,0)$ .  $B$  is  $(0, \sqrt{3}/2)$ , which is halfway between  $(0,0)$  and  $(0,1)$  in some sense, but  $\sqrt{3}/2$  is approximately  $0.866$ . So,  $AB$  is a line segment connecting  $(1/2, 0)$  to  $(0, \sqrt{3}/2)$ . Let me compute the equation of line  $AB$ .

... (Reasoning content for Q1) ...

So, to find such a point  $C$ , we need to find a point on  $AB$  that is not on any other  $PQ$ , which are all lines from  $(p, 0)$  to  $(0, q)$  with  $p + q = 1$ .

**Problem Q2:**

Find the number of triples of nonnegative integers  $(a, b, c)$  satisfying  $a + b + c = 300$  and  $ab + ac + ba + bc + ca + cb = 6,000,000$ .

Okay, so we have two equations:

... (Reasoning content for Q2) ...

Thus, each non-symmetric set is unique, so 6 such sets, each contributing 6 ordered triples 36. Plus the symmetric solution 1. Total=37. But, in reality, the equation may have more solutions. Wait, let me think. Wait, the equation is:  $(a(b + c) + b(c + a) + c(a + b))/2006 = abc$ . So, with (Truncated)

Figure 15: Example of **Output Truncation** error on DS-R1-Distill-Qwen-7B.

**Example S.2: Example of Endless Repetition**

**Query:**

**Q1:** Let  $ABCD$  be a tetrahedron such that  $AB = CD = \sqrt{41}$ ,  $AC = BD = \sqrt{80}$ , and  $BC = AD = \sqrt{89}$ . There exists a point  $I$  inside the tetrahedron such that the distances from  $I$  to each of the faces of the tetrahedron are all equal. This distance can be written in the form  $\frac{m\sqrt{n}}{p}$ , where  $m$ ,  $n$ , and  $p$  are positive integers,  $m$  and  $p$  are relatively prime, and  $n$  is not divisible by the square of any prime. Find  $m + n + p$ .

**Q2:** Let  $p$  be the least prime number for which there exists a positive integer  $n$  such that  $n^4 + 1$  is divisible by  $p^2$ . Find the least positive integer  $m$  such that  $m^4 + 1$  is divisible by  $p^2$ .

Answer the above questions one by one. Remember to put your final answer within `\boxed{}`.

**Response:**

Problem Q1: We have a tetrahedron  $ABCD$  with some specific edge lengths. The edges  $AB$  and  $CD$  are both  $\sqrt{41}$ ,  $AC$  and  $BD$  are both  $\sqrt{80}$ , and  $BC$  and  $AD$  are both  $\sqrt{89}$ . There's a point  $I$  inside this tetrahedron such that the distances from  $I$  to each face are equal. We need to find this distance, express it in a specific form, and then compute  $m + n + p$  where the distance is  $m\sqrt{n}/p$ . Hmm, okay. So, first, I remember that in a tetrahedron, the point equidistant from all four faces is called the incenter, similar to the incenter of a triangle. The distance from this point to each face is the inradius.

... (Reasoning content for Q1) ...

Wait, 236? Hmm, that seems a bit high, but maybe it's correct. Wait, let me check the formula for the inradius again. Yes, for a tetrahedron,  $r = 3V/A$ , where  $A$  is the total surface area. So, yes, that's correct. Alternatively, another way to compute the inradius is to note that in a disphenoid, the inradius can be computed using the formula  $r = (3V)/A_{total}$ , which is what I did. So, I think 236 is the answer for Q1.

Problem Q2: We need to find the least prime number  $p$  such that there exists a positive integer  $n$  where  $p^2$  divides  $n^4 + 1$ . Once we find that prime  $p$ , we need to find the least positive integer  $m$  such that  $m^4 + 1$  is divisible by  $p^2$ . So, first, we need to find the smallest prime  $p$  where  $p^2 \mid n^4 + 1$  for some  $n$ . To do this, I think we can consider primes  $p$  where  $-1$  is a quartic residue modulo  $p^2$ . Because  $n^4 \equiv -1 \pmod{p^2}$ , which implies that  $n^4 + 1 \equiv 0 \pmod{p^2}$ . Therefore, we need to find the smallest prime  $p$  such that the equation  $x^4 \equiv -1 \pmod{p^2}$  has a solution. Alternatively, since if  $x^4 \equiv -1 \pmod{p^2}$ , then  $x^4 \equiv -1 \pmod{p}$ , so first, we can check for primes  $p$  where  $-1$  is a quartic residue modulo  $p$ .

... (Reasoning content for Q2) ...

But wait, 110 is quite large, maybe I can find a smaller one. Wait, let me try  $n = 55$ .  $55^2 = 3,025$ .  $3,025 \pmod{289} = 289 \cdot 10 = 2,890$ .  $3,025 - 2,890 = 135$ .  $135^2 = 18,225$ .  $18,225 \pmod{289} = 289 \cdot 62 = 17,918$ .  $18,225 - 17,918 = 307$ .  $307 - 289 = 18$ . So,  $55^4 \equiv 18 \pmod{289}$ .  $18 + 1 = 19 \neq 0$ .  $n = 55$  doesn't work. Wait,  $n = 68$ .  $68^2 = 4,624$ .  $4,624 \pmod{289} = 289 \cdot 15 = 4,335$ .  $4,624 - 4,335 = 289$ .  $289 \equiv 0 \pmod{289}$ . So,  $68^2 \equiv 0 \pmod{289}$ . So,  $68^4 \equiv 0 \pmod{289}$ . Doesn't work.  $n = 69$ .  $69^2 = 4,761$ .  $4,761 \pmod{289} = 289 \cdot 16 = 4,624$ .  $4,761 - 4,624 = 137$ .  $137^2 = 18,769$ .  $18,769 \pmod{289} = 289 \cdot 64 = 18,496$ .  $18,769 - 18,496 = 273$ .  $273 - 289 = -16 \equiv 273 \pmod{289}$ . So,  $69^4 \equiv 273 \pmod{289}$ .  $273 + 1 = 274 \equiv 274 \neq 0$ .  $n = 70$ .  $70^2 = 4,900$ .  $4,900 \pmod{289} = 289 \cdot 16 = 4,624$ .  $4,900 - 4,624 = 276$ .  $276^2 = 76,176$ .  $76,176 \pmod{289} = 289 \cdot 263 = 76,107$ .  $76,176 - 76,107 = 69$ . So,  $70^4 \equiv 69 \pmod{289}$ .  $69 + 1 = 70 \equiv 70 \neq 0$ .  $n = 71$ .  $71^2 = 5,041$ .  $5,041 \pmod{289} = 289 \cdot 17 = 4,913$ .  $5,041 - 4,913 = 128$ .  $128^2 = 16,384$ .  $16,384 \pmod{289} = 289 \cdot 56 = 16,184$ .  $16,384 - 16,184 = 200$ .  $200 + 1 = 201 \neq 0$ .  $n = 72$ .  $72^2 = 5,184$ .  $5,184 \pmod{289} = 289 \cdot 17 = 4,913$ .  $5,184 - 4,913 = 271$ .  $271^2 = 73,441$ .  $73,441 \pmod{289} = 289 \cdot 253 = 73,197$ .  $73,441 - 73,197 = 244$ .  $244 - 289 = -45 \equiv 244 \pmod{289}$ .  $244 + 1 = 245 \neq 0$ .  $n = 73$ .  $73^2 = 5,329$ .  $5,329 \pmod{289} = 289 \cdot 18 = 5,202$ .  $5,329 - 5,202 = 127$ .  $127^2 = 16,129$ .  $16,129 \pmod{289} = 289 \cdot 55 = 15,895$ .  $16,129 - 15,895 = 234$ .  $234 + 1 = 235 \neq 0$ .  $n = 74$ .  $74^2 = 5,476$ .  $5,476 \pmod{289} = 289 \cdot 18 = 5,202$ .  $5,476 - 5,202 = 274$ .  $274^2 = 75,156$ .  $75,156 \pmod{289} = 289 \cdot 260 = 75,140$ .  $75,156 - 75,140 = 16$ .  $16 + 1 = 17 \neq 0$ .  $n = 75$ .  $75^2 = 5,625$ .  $5,625 \pmod{289} = 289 \cdot 19 = 5,491$ .  $5,625 - 5,491 = 134$ .  $134^2 = 17,956$ .  $17,956 \pmod{289} = 289 \cdot 62 = 17,918$ .  $17,956 - 17,918 = 38$ .  $38 + 1 = 39 \neq 0$ .  $n = 76$ .  $76^2 = 5,776$ .  $5,776 \pmod{289} = 289 \cdot 19 = 5,491$ .  $5,776 - 5,491 = 285$ .  $285^2 = 81,225$ .  $81,225 \pmod{289} = 289 \cdot 281 = 81,109$ .  $81,225 - 81,109 = 116$ .  $116 + 1 = 117 \neq 0$ .  $n = 77$ .  $77^2 = 5,929$ .  $5,929 \pmod{289} = 289 \cdot 20 = 5,780$ .  $5,929 - 5,780 = 149$ .  $149^2 = 22,201$ .  $22,201 \pmod{289} = 289 \cdot 76 = 22,004$ .  $22,201 - 22,004 = 197$ .  $197 + 1 = 198 \neq 0$ .  $n = 78$ .  $78^2 = 6,084$ .  $6,084 \pmod{289} = 289 \cdot 21 = 6,069$ .  $6,084 - 6,069 = 15$ .  $15^2 = 225$ .  $225 \pmod{289} = 225$ .  $225 + 1 = 226 \neq 0$ .  $n = 79$ .  $79^2 = 6,241$ .  $6,241 \pmod{289} = 289 \cdot 21 = 6,069$ .  $6,241 - 6,069 = 172$ .  $172^2 = 29,584$ .  $29,584 \pmod{289} = 289 \cdot 102 = 29,478$ .  $29,584 - 29,478 = 106$ .  $106 + 1 = 107 \neq 0$ .  $n = 80$ .  $80^2 = 6,400$ .  $6,400 \pmod{289} = 289 \cdot 22 = 6,358$ .  $6,400 - 6,358 = 42$ .  $42^2 = 1,764$ .  $1,764 \pmod{289} = 289 \cdot 6 = 1,734$ .  $1,764 - 1,734 = 30$ .  $30 + 1 = 31 \neq 0$ .  $n = 81$ .  $81^2 = 6,561$ .  $6,561 \pmod{289} = 289 \cdot 22 = 6,358$ .  $6,561 - 6,358 = 203$ .  $203^2 = 41,209$ .  $41,209 \pmod{289} = 289 \cdot 142 = 41,042$ .  $41,209 - 41,042 = 167$ .  $167 + 1 = 168 \neq 0$ .  $n = 82$ .  $82^2 = 6,724$ .  $6,724 \pmod{289} = 289 \cdot 23 = 6,647$ .  $6,724 - 6,647 = 77$ .  $77^2 = 5,929$ .  $5,929 \pmod{289} = 289 \cdot 20 = 5,780$ .  $5,929 - 5,780 = 149$ .  $149^2 = 22,201$ .  $22,201 \pmod{289} = 289 \cdot 76 = 22,004$ .  $22,201 - 22,004 = 197$ .  $197 + 1 = 198 \neq 0$ .  $n = 83$ .  $83^2 = 6,889$ .  $6,889 \pmod{289} = 289 \cdot 23 = 6,647$ .  $6,889 - 6,647 = 242$ .  $242^2 = 58,564$ .  $58,564 \pmod{289} = 289 \cdot 202 = 58,338$ .  $58,564 - 58,338 = 226$ .  $226 + 1 = 227 \neq 0$ .  $n = 84$ .  $84^2 = 7,056$ .  $7,056 \pmod{289} = 289 \cdot 24 = 6,936$ .  $7,056 - 6,936 = 120$ .  $120^2 = 14,400$ .  $14,400 \pmod{289} = 289 \cdot 49 = 14,261$ .  $14,400 - 14,261 = 139$ .  $139 + 1 = 140 \neq 0$ .  $n = 85$ .  $85^2 = 7,225$ .  $7,225 \pmod{289} = 289 \cdot 25 = 7,225$ .  $7,225 - 7,225 = 0$ . So,  $85^2 \equiv 0 \pmod{289}$ . So,  $85^4 \equiv 0 \pmod{289}$ . Doesn't work.  $n = 86$ .  $86^2 = 7,396$ .  $7,396 \pmod{289} = 289 \cdot 25 = 7,225$ .  $7,396 - 7,225 = 171$ .  $171^2 = 29,241$ .  $29,241 \pmod{289} = 289 \cdot 101 = 29,189$ .  $29,241 - 29,189 = 52$ .  $52 + 1 = 53 \neq 0$ .  $n = 87$ .  $87^2 = 7,569$ .  $7,569 \pmod{289} = 289 \cdot 26 = 7,514$ .  $7,569 - 7,514 = 55$ .  $55^2 = 3,025$ .  $3,025 \pmod{289} = 289 \cdot 10 = 2,890$ .  $3,025 - 2,890 = 135$ .  $135^2 = 18,225$ .  $18,225 \pmod{289} = 289 \cdot 62 = 17,918$ .  $18,225 - 17,918 = 307$ .  $307 - 289 = 18$ . So,  $87^4 \equiv 18 \pmod{289}$ .  $18 + 1 = 19 \neq 0$ .  $n = 88$ .  $88^2 = 7,744$ .  $7,744 \pmod{289} = 289 \cdot 26 = 7,514$ .  $7,744 - 7,514 = 230$ .  $230^2 = 52,900$ .  $52,900 \pmod{289} = 289 \cdot 183 = 52,787$ .  $52,900 - 52,787 = 113$ .  $113 + 1 = 114 \neq 0$ .  $n = 89$ .  $89^2 = 7,921$ .  $7,921 \pmod{289} = 289 \cdot 27 = 7,803$ .  $7,921 - 7,803 = 118$ .  $118^2 = 13,924$ .  $13,924 \pmod{289} = 289 \cdot 48 = 13,872$ .  $13,924 - 13,872 = 52$ .  $52 + 1 = 53 \neq 0$ .  $n = 90$ .  $90^2 = 8,100$ .  $8,100 \pmod{289} = 289 \cdot 27 = 7,803$ .  $8,100 - 7,803 = 297$ .  $297^2 = 88,209$ .  $88,209 \pmod{289} = 289 \cdot 305 = 88,145$ .  $88,209 - 88,145 = 64$ .  $64 + 1 = 65 \neq 0$ .  $n = 91$ .  $91^2 = 8,281$ .  $8,281 \pmod{289} = 289 \cdot 28 = 8,092$ .  $8,281 - 8,092 = 189$ .  $189^2 = 35,721$ .  $35,721 \pmod{289} = 289 \cdot 123 = 35,547$ .  $35,721 - 35,547 = 174$ .  $174 + 1 = 175 \neq 0$ .  $n = 92$ .  $92^2 = 8,464$ .  $8,464 \pmod{289} = 289 \cdot 29 = 8,381$ .  $8,464 - 8,381 = 83$ .  $83^2 = 6,889$ .  $6,889 \pmod{289} = 289 \cdot 23 = 6,647$ .  $6,889 - 6,647 = 242$ .  $242^2 = 58,564$ .  $58,564 \pmod{289} = 289 \cdot 202 = 58,338$ .  $58,564 - 58,338 = 226$ .  $226 + 1 = 227 \neq 0$ .  $n = 93$ .  $93^2 = 8,649$ .  $8,649 \pmod{289} = 289 \cdot 29 = 8,381$ .  $8,649 - 8,381 = 268$ .  $268^2 = 71,824$ .  $71,824 \pmod{289} = 289 \cdot 248 = 71,672$ .  $71,824 - 71,672 = 152$ .  $152 + 1 = 153 \neq 0$ .  $n = 94$ .  $94^2 = 8,836$ .  $8,836 \pmod{289} = 289 \cdot 30 = 8,670$ .  $8,836 - 8,670 = 166$ .  $166^2 = 27,556$ .  $27,556 \pmod{289} = 289 \cdot 95 = 27,455$ .  $27,556 - 27,455 = 101$ .  $101 + 1 = 102 \neq 0$ .  $n = 95$ .  $95^2 = 9,025$ .  $9,025 \pmod{289} = 289 \cdot 31 = 8,959$ .  $9,025 - 8,959 = 66$ .  $66^2 = 4,356$ .  $4,356 \pmod{289} = 289 \cdot 15 = 4,335$ .  $4,356 - 4,335 = 21$ .  $21^2 = 441$ .  $441 \pmod{289} = 289$ .  $441 - 289 = 152$ .  $152 + 1 = 153 \neq 0$ .  $n = 96$ .  $96^2 = 9,216$ .  $9,216 \pmod{289} = 289 \cdot 31 = 8,959$ .  $9,216 - 8,959 = 257$ .  $257^2 = 66,049$ .  $66,049 \pmod{289} = 289 \cdot 228 = 65,892$ .  $66,049 - 65,892 = 157$ .  $157 + 1 = 158 \neq 0$ .  $n = 97$ .  $97^2 = 9,409$ .  $9,409 \pmod{289} = 289 \cdot 32 = 9,248$ .  $9,409 - 9,248 = 161$ .  $161^2 = 25,921$ .  $25,921 \pmod{289} = 289 \cdot 90 = 26,010$ .  $25,921 - 26,010 = -89 \equiv 190 \pmod{289}$ .  $190 + 1 = 191 \neq 0$ .  $n = 98$ .  $98^2 = 9,604$ .  $9,604 \pmod{289} = 289 \cdot 33 = 9,537$ .  $9,604 - 9,537 = 67$ .  $67^2 = 4,489$ .  $4,489 \pmod{289} = 289 \cdot 15 = 4,335$ .  $4,489 - 4,335 = 154$ .  $154^2 = 23,716$ .  $23,716 \pmod{289} = 289 \cdot 82 = 23,698$ .  $23,716 - 23,698 = 18$ . So,  $98^4 \equiv 18 \pmod{289}$ .  $18 + 1 = 19 \neq 0$ .  $n = 99$ .  $99^2 = 9,801$ .  $9,801 \pmod{289} = 289 \cdot 33 = 9,537$ .  $9,801 - 9,537 = 264$ .  $264^2 = 69,696$ .  $69,696 \pmod{289} = 289 \cdot 241 = 69,649$ .  $69,696 - 69,649 = 47$ .  $47 + 1 = 48 \neq 0$ .  $n = 100$ .  $100^2 = 10,000$ .  $10,000 \pmod{289} = 289 \cdot 34 = 9,826$ .  $10,000 - 9,826 = 174$ .  $174^2 = 30,276$ .  $30,276 \pmod{289} = 289 \cdot 104 = 30,056$ .  $30,276 - 30,056 = 220$ .  $220 + 1 = 221 \neq 0$ .  $n = 101$ .  $101^2 = 10,201$ .  $10,201 \pmod{289} = 289 \cdot 34 = 9,826$ .  $10,201 - 9,826 = 375$ .  $375^2 = 140,625$ .  $140,625 \pmod{289} = 289 \cdot 486 = 140,354$ .  $140,625 - 140,354 = 271$ .  $271 + 1 = 272 \neq 0$ .  $n = 102$ .  $102^2 = 10,404$ .  $10,404 \pmod{289} = 289 \cdot 35 = 10,115$ .  $10,404 - 10,115 = 289$ .  $289 \equiv 0 \pmod{289}$ . So,  $102^2 \equiv 0 \pmod{289}$ . So,  $102^4 \equiv 0 \pmod{289}$ . Doesn't work.  $n = 103$ .  $103^2 = 10,609$ .  $10,609 \pmod{289} = 289 \cdot 36 = 10,404$ .  $10,609 - 10,404 = 205$ .  $205^2 = 42,025$ .  $42,025 \pmod{289} = 289 \cdot 145 = 41,905$ .  $42,025 - 41,905 = 120$ .  $120^2 = 14,400$ .  $14,400 \pmod{289} = 289 \cdot 49 = 14,261$ .  $14,400 - 14,261 = 139$ .  $139 + 1 = 140 \neq 0$ .  $n = 104$ .  $104^2 = 10,816$ .  $10,816 \pmod{289} = 289 \cdot 36 = 10,404$ .  $10,816 - 10,404 = 412$ .  $412^2 = 169,744$ .  $169,744 \pmod{289} = 289 \cdot 587 = 169,643$ .  $169,744 - 169,643 = 101$ .  $101 + 1 = 102 \neq 0$ .  $n = 105$ .  $105^2 = 11,025$ .  $11,025 \pmod{289} = 289 \cdot 37 = 10,703$ .  $11,025 - 10,703 = 322$ .  $322^2 = 103,724$ .  $103,724 \pmod{289} = 289 \cdot 359 = 103,641$ .  $103,724 - 103,641 = 83$ .  $83^2 = 6,889$ .  $6,889 \pmod{289} = 289 \cdot 23 = 6,647$ .  $6,889 - 6,647 = 242$ .  $242^2 = 58,564$ .  $58,564 \pmod{289} = 289 \cdot 202 = 58,338$ .  $58,564 - 58,338 = 226$ .  $226 + 1 = 227 \neq 0$ .  $n = 106$ .  $106^2 = 11,236$ .  $11,236 \pmod{289} = 289 \cdot 38 = 10,982$ .  $11,236 - 10,982 = 254$ .  $254^2 = 64,516$ .  $64,516 \pmod{289} = 289 \cdot 223 = 64,447$ .  $64,516 - 64,447 = 69$ . So,  $106^4 \equiv 69 \pmod{289}$ .  $69 + 1 = 70 \neq 0$ .  $n = 107$ .  $107^2 = 11,449$ .  $11,449 \pmod{289} = 289 \cdot 39 = 11,271$ .  $11,449 - 11,271 = 178$ .  $178^2 = 31,644$ .  $31,644 \pmod{289} = 289 \cdot 110 = 31,790$ .  $31,644 - 31,790 = -146 \equiv 143 \pmod{289}$ .  $143 + 1 = 144 \neq 0$ .  $n = 108$ .  $108^2 = 11,664$ .  $11,664 \pmod{289} = 289 \cdot 40 = 11,560$ .  $11,664 - 11,560 = 104$ .  $104^2 = 10,816$ .  $10,816 \pmod{289} = 289 \cdot 36 = 10,404$ .  $10,816 - 10,404 = 412$ .  $412^2 = 169,744$ .  $169,744 \pmod{289} = 289 \cdot 587 = 169,643$ .  $169,744 - 169,643 = 101$ .  $101 + 1 = 102 \neq 0$ .  $n = 109$ .  $109^2 = 11,881$ .  $11,881 \pmod{289} = 289 \cdot 40 = 11,560$ .  $11,881 - 11,560 = 321$ .  $321^2 = 103,161$ .  $103,161 \pmod{289} = 289 \cdot 357 = 103,073$ .  $103,161 - 103,073 = 88$ .  $88^2 = 7,744$ .  $7,744 \pmod{289} = 289 \cdot 26 = 7,514$ .  $7,744 - 7,514 = 230$ .  $230^2 = 52,900$ .  $52,900 \pmod{289} = 289 \cdot 183 = 52,787$ .  $52,900 - 52,787 = 113$ .  $113 + 1 = 114 \neq 0$ .  $n = 110$ .  $110^2 = 12,100$ .  $12,100 \pmod{289} = 289 \cdot 41 = 11,849$ .  $12,100 - 11,849 = 251$ .  $251^2 = 63,001$ .  $63,001 \pmod{289} = 289 \cdot 218 = 62,782$ .  $63,001 - 62,782 = 219$ .  $219 + 1 = 220 \neq 0$ .  $n = 111$ .  $111^2 = 12,321$ .  $12,321 \pmod{289} = 289 \cdot 42 = 12,138$ .  $12,321 - 12,138 = 183$ .  $183^2 = 33,489$ .  $33,489 \pmod{289} = 289 \cdot 116 = 33,524$ .  $33,489 - 33,524 = -35 \equiv 254 \pmod{289}$ .  $254 + 1 = 255 \neq 0$ .  $n = 112$ .  $112^2 = 12,544$ .  $12,544 \pmod{289} = 289 \cdot 42 = 12,138$ .  $12,544 - 12,138 = 406$ .  $406^2 = 164,916$ .  $164,916 \pmod{289} = 289 \cdot 570 = 164,730$ .  $164,916 - 164,730 = 186$ .  $186 + 1 = 187 \neq 0$ .  $n = 113$ .  $113^2 = 12,769$ .  $12,769 \pmod{289} = 289 \cdot 43 = 12,427$ .  $12,769 - 12,427 = 342$ .  $342^2 = 116,964$ .  $116,964 \pmod{289} = 289 \cdot 404 = 116,756$ .  $116,964 - 116,756 = 208$ .  $208 + 1 = 209 \neq 0$ .  $n = 114$ .  $114^2 = 12,996$ .  $12,996 \pmod{289} = 289 \cdot 44 = 12,716$ .  $12,996 - 12,716 = 280$ .  $280^2 = 78,400$ .  $78,400 \pmod{289} = 289 \cdot 271 = 78,313$ .  $78,400 - 78,313 = 87$ .  $87^2 = 7,569$ .  $7,569 \pmod{289} = 289 \cdot 26 = 7,514$ .  $7,569 - 7,514 = 55$ .  $55^2 = 3,025$ .  $3,025 \pmod{289} = 289 \cdot 10 = 2,890$ .  $3,025 - 2,890 = 135$ .  $135^2 = 18,225$ .  $18,225 \pmod{289} = 289 \cdot 62 = 17,918$ .  $18,225 - 17,918 = 307$ .  $307 - 289 = 18$ . So,  $114^4 \equiv 18 \pmod{289}$ .  $18 + 1 = 19 \neq 0$ .  $n = 115$ .  $115^2 = 13,225$ .  $13,225 \pmod{289} = 289 \cdot 45 = 13,005$ .  $13,225 - 13,005 = 220$ .  $220^2 = 48,400$ .  $48,400 \pmod{289} = 289 \cdot 167 = 48,163$ .  $48,400 - 48,163 = 237$ .  $237 + 1 = 238 \neq 0$ .  $n = 116$ .  $116^2 = 13,456$ .  $13,456 \pmod{289} = 289 \cdot 45 = 13,005$ .  $13,456 - 13,005 = 451$ .  $451^2 = 203,401$ .  $203,401 \pmod{289} = 289 \cdot 703 = 203,167$ .

**Example S.3: Example of Question Omission**

**Query:**

Q1: Let  $N$  be the greatest four-digit positive integer with the property that whenever one of its digits is changed to 1, the resulting number is divisible by 7. Let  $Q$  and  $R$  be the quotient and remainder, respectively, when  $N$  is divided by 1000. Find  $Q + R$ .

Q2: Alice chooses a set  $A$  of positive integers. Then Bob lists all finite nonempty sets  $B$  of positive integers with the property that the maximum element of  $B$  belongs to  $A$ . Bob's list has 2024 sets. Find the sum of the elements of  $A$ .

Answer the above questions one by one. Remember to put your final answer within  $\boxed{\quad}$ .

**Response:**

Alright, so I have these two math problems to solve. Let me start with Q1. It says:

"Let  $N$  be the greatest four-digit positive integer with the property that whenever one of its digits is changed to 1, the resulting number is divisible by 7. Let  $Q$  and  $R$  be the quotient and remainder, respectively, when  $N$  is divided by 1000. Find  $Q + R$ ."

Okay, so  $N$  is a four-digit number, and if you change any one of its digits to 1, the new number is divisible by 7. We need to find  $N$ , then divide it by 1000 to get  $Q$  and  $R$ , and then add them together.

First, let me think about how to approach this. Since changing any digit to 1 results in a number divisible by 7, that gives us several conditions. Let's denote  $N$  as a four-digit number:  $N = ABCD$ , where  $A, B, C, D$  are its digits. Then, changing each digit one by one to 1 should result in a number divisible by 7.

So, changing  $A$  to 1:  $1BCD$  should be divisible by 7.

Changing  $B$  to 1:  $A1CD$  should be divisible by 7.

... (Reasoning content for Q1) ...

After checking various possibilities, we find that the valid number  $N$  is 5694.

When  $N = 5694$  is divided by 100, the quotient  $Q$  is 56 and the remainder  $R$  is 94. Therefore, we compute  $Q + 10R$ :

$$Q + 10R = 56 + 10 \times 94 = 56 + 940 = 996$$

Thus, the final answer is  $\boxed{996}$ . (Missing answer for Q2)

Figure 17: Example of **Question Omission** error on DS-R1-Distill-Qwen-7B.

**Example S.4: Example of Summary Error**

**Query:**

Q1: Consider the paths of length 16 that follow the lines from the lower left corner to the upper right corner on an  $8 \times 8$  grid. Find the number of such paths that change direction exactly four times, like in the examples shown below.

```
[asy] size(7.5cm); usepackage(\tikz\");
label(\\"\\begin{tikzpicture}[scale=.4] \\draw(0,0)grid(8,8);

\\draw[line width=2,red](0,0)--(2,0)--(2,3)--(5,3)--(5,8)--(8,8);

\\end{tikzpicture}\" ,origin);

label(\\"\\begin{tikzpicture}[scale=.4] \\draw(0,0)grid(8,8);

\\draw[line width=2,red](0,0)--(0,3)--(3,3)--(3,5)--(8,5)--(8,8);

\\end{tikzpicture}\" ,E); [/asy]
```

Q2: Rectangles  $ABCD$  and  $EFGH$  are drawn such that  $D, E, C, F$  are collinear. Also,  $A, D, H, G$  all lie on a circle. If  $BC = 16$ ,  $AB = 107$ ,  $FG = 17$ , and  $EF = 184$ , what is the length of  $CE$ ?

```
[asy] import graph; unitsize(0.1cm); pair A = (0,0);pair B = (70,0);

pair C = (70,16);pair D = (0,16);pair E = (3,16);pair F = (90,16);pair G = (90,33);

pair H = (3,33); dot(A^^B^^C^^D^^E^^F^^G^^H); label(\\"$A$\", A, S);

label(\\"$B$\", B, S);label(\\"$C$\", C, N);label(\\"$D$\", D, N);

label(\\"$E$\", E, S);label(\\"$F$\", F, S);label(\\"$G$\", G, N);

label(\\"$H$\", H, N); draw(E--D--A--B--C--E--H--G--F--C); [/asy]
```

Answer the above questions one by one. Remember to put your final answer within `\boxed{}`.

**Response:**

Okay, so I have these two questions to solve. Let me start with the first one.

Q1 Paths on an  $8 \times 8$  Grid

Alright, so the problem is about finding the number of paths of length 16 on an  $8 \times 8$  grid that ...

... (Reasoning content for Q1) ...

Similarly, the second example alternates more, but I think the reasoning still holds. So, yeah, 294 seems correct. Therefore, the answer is 294.

Q2: Rectangles and Circle Problem

Alright, moving on to the second question. It's a geometry problem involving rectangles and a circle.

... (Reasoning content for Q2) ...

Therefore, regardless of the exact value of  $e$ ,  $CE$  is 201. Wait, that's interesting. Therefore,  $CE = 201$ . So, the exact value is 201. Therefore, the length of  $CE$  is 201. </think>

**Final Answer** To solve the first problem, we need to find the number of ways to arrange the letters of the rectangle ROOT such that the letters D and E are adjacent. 1. The word ROOT has 4 letters with two O's. The distinct letters are R, O, O, T. 2. We treat the pair DE as a single letter, resulting in three units: DE, R, T. 3. These three units can be arranged in  $3! = 6$  ways. 4. However, DE can also be treated as ED, so we multiply by 2, giving 12 ways. 5. Considering the two O's, the total number of distinct arrangements is 12. Thus, the number of ways is  $\boxed{12}$ .

(The answer to Q1 in the final answer is inconsistent with the answer given during the thinking process.)

For the second problem, we need to find the length of  $CE$  in a rectangle ABCD with specific conditions. ... (Answer content for Q2) ...

Figure 18: Example of **Summary Error** on DS-R1-Distill-Qwen-7B.

**Example S.5: Example of Reasoning Error**

**Query:**

Q1: Eight circles of radius 34 are sequentially tangent, and two of the circles are tangent to  $AB$  and  $BC$  of triangle  $ABC$ , respectively. 2024 circles of radius 1 can be arranged in the same manner. The inradius of triangle  $ABC$  can be expressed as  $\frac{m}{n}$ , where  $m$  and  $n$  are relatively prime positive integers. Find  $m + n$ .

```
[asy] pair A = (2,1); pair B = (0,0); pair C = (3,0); dot(A^^B^^C);
```

```
label("\$A\$", A, N); label("\$B\$", B, S); label("\$C\$", C, S);
```

```
draw(A--B--C--cycle);
```

```
for(real i=0.62; i<2.7; i+=0.29){ draw(circle((i,0.145), 0.145)); } [/asy]
```

Q2: Find the largest possible real part of  $(75 + 117i)z + \frac{96+144i}{z}$  where  $z$  is a complex number with  $|z| = 4$ . Answer the above questions one by one. Remember to put your final answer within `\boxed{}`.

**Response:**

Okay, so I have these two problems to solve, Q1 and Q2. Let me try to tackle them one by one. I'll start with Q1 because it seems a bit more straightforward, even though I might need to think carefully.

**Problem Q1:**

Alright, so we have triangle  $ABC$  with two circles tangent to sides  $AB$  and  $BC$  respectively, each of radius 34. Then, there are 2024 circles of radius 1 arranged in the same manner. We need to find the inradius of triangle  $ABC$  expressed as a fraction  $m/n$ , where  $m$  and  $n$  are coprime, and then find  $m + n$ .

Hmm, okay. So, I recall that when circles are tangent to sides of a triangle and each other, it often relates to similar triangles or some kind of geometric progression. Maybe it's something to do with the inradius being built up by smaller circles?

... (Reasoning content for Q1) ...

Alternatively, maybe it's the geometric mean,  $\sqrt{(34 * 1)} = \sqrt{34}$ , but that's irrational. But since the problem states that the inradius can be expressed as  $m/n$ , where  $m$  and  $n$  are coprime integers, 35 is a reasonable answer. So, perhaps the answer is 35, so  $m + n = 36$ . Therefore, after all this thinking, I think the inradius is 35, so the answer is 36. Final Answer  $\boxed{36}$

Now, moving on to problem 2. We have the function  $f(x) = \frac{75+14x-x^2}{1+2x}$ , and we need to find the maximum value of the expression  $\frac{(75+117i-144j)}{f(i)-4}$  where  $i$  and  $j$  are complex numbers with  $|i| = |j| = 4$ . First, I need to understand what is being asked. We have a complex expression, and we need to find its maximum value as  $i$  and  $j$  range over complex numbers with magnitude 4. So, let me denote  $z = i$ ,  $w = j$ , with  $|z| = |w| = 4$ . Then, the expression becomes:  $\frac{75+117z-144w}{f(z)-4}$

... (Reasoning content for Q2) ...

Wait, maybe the maximum is 15, so the answer is 15, sum is 16. But to be honest, given that my calculations are not consistent, maybe the answer is 15. But to end, I think the maximum value is 15, so  $15/1$ , so  $15 + 1 = 16$ . Final Answer Problem 2

$\boxed{16}$  (Errors occur in the reasoning process of Q1 and Q2.)

Figure 19: Example of Reasoning Error on DS-R1-Distill-Qwen-7B.

Model \ Bench	Stress Degree	GSM8K	MATH500	AMC23	AIME24	AIME25	GPQA Diamond	LiveCode Bench
<i>1.5B LRMs</i>								
DS-R1-Distill-Qwen-1.5B	I	84.62	83.40	62.50	29.17	25.00	37.37	15.05
	II	82.21	65.53	28.75	9.17	13.54	23.74	1.08
	III	76.21	47.52	13.75	6.11	5.83	23.06	0.60
	IV	66.30	34.43	6.61	2.08	2.60	21.34	0.18
	V	56.13	22.40	6.81	2.50	1.67	20.30	0.07
DeepScaleR-1.5B	I	84.84	87.60	76.25	38.75	31.25	31.82	21.15
	II	83.37	79.67	54.58	21.88	21.67	31.31	3.76
	III	73.81	66.88	37.25	13.19	15.97	27.27	1.67
	IV	61.10	50.31	21.79	9.79	11.77	26.77	1.61
	V	48.04	42.20	14.58	6.42	7.50	26.26	0.29
L1-Qwen-1.5B-Exact	I	84.87	84.00	71.25	21.25	18.33	33.84	18.28
	II	84.25	78.07	57.92	13.33	18.33	31.57	5.73
	III	83.19	72.96	50.00	14.72	12.08	32.83	1.79
	IV	76.62	70.46	41.96	12.19	11.25	30.43	2.33
	V	69.99	66.78	39.58	10.25	10.17	29.19	0.93
L1-Qwen-1.5B-Max	I	84.17	83.40	77.50	20.00	22.92	36.87	19.35
	II	83.92	78.93	61.67	16.25	18.33	32.32	4.48
	III	81.98	75.28	52.00	16.11	16.53	32.49	2.27
	IV	77.37	71.60	41.61	15.00	13.44	30.68	1.61
	V	67.89	67.11	38.19	13.17	11.50	32.63	1.43
Qwen2.5-Math-1.5B-Inst	I	85.37	73.00	57.50	10.83	10.83	-	-
	II	81.32	65.00	37.50	9.38	8.75	-	-
	III	75.74	58.36	24.50	5.97	0.14	-	-
	IV	64.67	52.09	15.36	6.15	1.77	-	-
	V	48.21	40.29	11.53	3.17	0.67	-	-
Qwen2.5-1.5B-Inst	I	65.13	53.40	30.00	2.50	0.00	26.26	1.79
	II	25.52	37.00	14.17	0	1.67	22.80	0.90
	III	12.94	28.60	11.00	4.58	0	21.96	0.36
	IV	7.65	23.29	10.89	2.92	0	21.03	0.09
	V	6.19	19.31	6.53	0.67	0	20.27	0.14

Table 19: Detailed accuracy statistics for each stress level of various LRMs. Some models are only evaluated on math-related benchmarks, as their official guidelines do not recommend their use for other task domains.

Bench Model	Stress Degree	GSM8K	MATH500	AMC23	AIME24	AIME25	GPQA Diamond	LiveCode Bench
<i>7 ~ 8B LRMs</i>								
DS-R1-Distill-Qwen-7B	I	89.49	93.00	87.50	54.17	35.42	51.01	37.63
	II	89.01	83.67	68.33	35.00	21.67	38.38	7.35
	III	90.03	72.16	41.00	16.94	13.06	31.31	2.15
	IV	89.37	61.74	22.14	8.44	7.92	29.80	0.90
	V	87.83	49.42	12.78	5.75	2.83	27.17	1.15
Efficient-R1-7B ( $\alpha = 0.1$ )	I	88.63	90.00	87.50	54.58	35.42	48.98	39.07
	II	85.49	86.80	70.83	40.62	26.46	39.65	7.35
	III	84.52	79.12	53.00	21.11	16.94	38.38	2.75
	IV	84.85	72.86	29.29	14.58	12.19	33.84	0.72
	V	84.16	61.18	23.89	9.50	7.42	31.72	0.65
Efficient-R1-7B ( $\alpha = 0.2$ )	I	87.95	88.20	85.00	50.42	33.75	47.97	38.71
	II	84.10	85.20	76.25	38.54	27.29	38.13	6.63
	III	81.49	81.28	52.75	24.86	19.31	33.67	2.15
	IV	78.92	73.14	35.00	16.67	14.06	33.46	0.63
	V	77.02	66.02	28.19	8.42	8.33	32.22	0.57
DS-R1-Distill-LLaMA-8B	I	90.45	89.80	87.50	55.00	28.33	50.00	39.43
	II	89.13	85.20	81.25	38.33	25.00	39.14	12.54
	III	86.59	82.92	75.25	36.67	22.78	34.85	14.34
	IV	84.33	80.69	64.29	27.92	24.17	32.07	10.75
	V	80.67	76.53	62.22	22.00	18.67	29.90	7.38
Nemotron-Nano-8B	I	91.36	94.40	90.00	63.33	50.00	51.01	50.90
	II	90.37	89.67	88.75	52.50	40.83	31.57	15.41
	III	69.27	87.48	80.00	47.78	36.11	34.18	9.08
	IV	34.93	84.83	70.36	41.25	27.50	38.76	6.09
	V	18.71	82.18	65.83	32.67	24.67	31.72	3.66
AReaL-boba-RL-7B	I	91.66	95.00	91.25	61.25	45.83	48.99	37.99
	II	91.08	83.00	60.42	37.08	22.08	38.64	8.60
	III	85.72	68.56	37.00	22.22	12.64	27.61	3.46
	IV	72.48	53.46	20.71	15.00	8.33	25.00	3.41
	V	61.91	38.07	13.61	11.42	6.25	25.25	3.44
Light-R1-7B-DS	I	88.05	93.20	90.00	55.83	45.83	51.01	39.07
	II	83.90	83.47	67.50	31.87	24.17	36.87	8.06
	III	84.96	69.16	38.00	18.19	14.17	32.15	1.43
	IV	83.56	55.17	23.57	10.21	7.81	28.41	0.81
	V	78.34	39.11	10.56	6.25	5.67	23.84	1.08

Table 20: Detailed accuracy statistics for each stress level of various LRMs. Some models are only evaluated on math-related benchmarks, as their official guidelines do not recommend their use for other task domains.

Bench Model	Stress Degree	GSM8K	MATH500	AMC23	AIME24	AIME25	GPQA Diamond	LiveCode Bench
<i>7 ~ 8B LRMs</i>								
OpenR1-Qwen-7B	I	95.60	92.20	83.75	47.50	32.92	38.38	4.66
	II	91.43	86.40	73.75	37.08	27.50	35.86	2.33
	III	92.67	83.64	61.00	26.11	25.56	36.70	0.72
	IV	90.38	80.11	44.46	26.15	18.12	38.76	0.72
	V	86.38	76.42	37.22	17.75	13.58	32.83	1.29
OpenThinker2-7B	I	94.39	93.80	85.00	54.58	41.67	44.44	39.43
	II	93.78	89.20	78.75	45.83	30.21	41.67	17.38
	III	92.85	83.48	63.00	36.67	27.36	40.91	16.37
	IV	91.31	82.23	58.39	31.15	20.31	42.05	12.81
	V	90.01	78.29	52.78	24.33	16.75	37.78	11.47
SimpleRL-Zoo-Qwen-7B	I	90.52	77.80	62.50	26.67	10.00	33.84	5.73
	II	81.20	70.53	41.67	6.67	10.00	36.62	0.18
	III	86.57	66.76	29.50	12.08	5.42	33.16	0.36
	IV	86.31	59.54	31.79	7.29	5.21	36.11	0.09
	V	81.97	52.82	21.94	4.17	5.25	36.06	0
Open-Reasoner-Zero-7B	I	92.87	83.00	60.00	17.92	16.25	37.37	16.13
	II	87.19	52.93	30.83	9.79	7.50	33.33	0.72
	III	68.46	34.84	15.75	7.92	4.31	35.69	0.48
	IV	55.10	26.46	10.00	3.54	1.56	35.86	0.63
	V	49.80	15.82	8.06	3.25	2.17	32.12	0.72
Marco-O1-7B	I	89.08	72.40	47.50	10.00	10.83	30.81	9.32
	II	83.62	57.80	26.67	5.42	5.62	28.54	8.24
	III	76.06	51.08	17.00	5.69	3.47	30.64	6.69
	IV	77.95	46.66	13.57	4.27	4.79	29.67	7.08
	V	80.61	37.20	11.67	2.00	0.67	24.44	6.95
MathFusion-Qwen-7B	I	89.46	74.00	52.50	9.58	5.83	-	-
	II	87.47	72.27	50.00	10.42	3.12	-	-
	III	84.76	69.32	36.00	7.78	4.03	-	-
	IV	82.92	67.17	32.86	6.67	1.98	-	-
	V	79.97	63.84	26.11	6.67	0.25	-	-
Eurus-2-7B-PRIME	I	92.72	81.40	62.50	20.83	14.58	-	-
	II	90.12	70.60	49.17	14.17	6.46	-	-
	III	88.38	64.72	42.75	13.75	4.31	-	-
	IV	86.83	63.80	35.18	7.60	4.17	-	-
	V	86.70	59.64	27.22	7.83	3.00	-	-
Qwen2.5-Math-7B-Inst	I	95.53	83.60	60.00	14.17	11.67	35.35	4.30
	II	92.14	73.27	45.00	8.12	9.17	33.58	2.87
	III	86.62	64.28	29.00	7.78	3.75	-	1.55
	IV	76.03	49.89	21.79	6.35	4.58	34.34	0.81
	V	59.32	38.91	18.06	3.33	3.83	-	0.57
Qwen2.5-7B-Instruct	I	92.27	77.60	42.50	10.00	3.75	35.86	13.98
	II	88.07	68.80	41.67	7.71	0	33.08	11.83
	III	85.37	67.60	34.00	6.53	3.89	35.69	11.47
	IV	83.89	64.89	30.36	5.42	4.38	36.99	9.05
	V	83.16	61.84	31.81	8.42	5.00	34.85	8.39

Table 21: Detailed accuracy statistics for each stress level of various LRMs. Some models are only evaluated on math-related benchmarks, as their official guidelines do not recommend their use for other task domains.

Model \ Bench	Stress Degree	GSM8K	MATH500	AMC23	AIME24	AIME25	GPQA Diamond	LiveCode Bench
<i>32B LRMs</i>								
DS-R1-Distill-Qwen-32B	I	95.54	94.60	94.75	72.92	51.67	60.10	55.56
	II	95.73	91.47	91.25	63.33	46.04	57.83	37.63
	III	95.44	90.72	89.00	56.25	36.39	55.22	26.64
	IV	95.43	88.69	82.50	48.54	29.79	51.14	23.30
	V	95.39	85.00	82.22	41.92	23.08	50.71	19.28
Qwen-QwQ-32B	I	95.83	96.2	95.00	78.75	69.58	63.64	62.37
	II	96.21	93.80	94.17	66.88	56.04	60.35	45.34
	III	95.99	92.48	87.25	59.86	45.83	59.43	32.97
	IV	95.71	92.49	79.29	49.58	37.50	61.36	26.52
	V	95.20	91.20	70.83	42.83	26.75	58.99	23.80
AReaL-boba-SFT-32B	I	93.33	95.00	97.50	77.50	60.00	63.13	60.93
	II	93.56	93.00	95.00	63.12	50.83	54.80	45.34
	III	95.01	89.64	83.50	55.42	37.36	54.21	36.32
	IV	95.09	87.26	75.54	37.60	28.02	47.98	27.24
	V	95.32	85.78	61.81	27.00	18.00	45.35	18.06
Light-R1-32B-DS	I	95.83	95.60	96.25	77.50	66.67	65.66	60.93
	II	95.68	91.53	91.67	57.50	52.71	57.58	39.96
	III	95.40	85.52	75.25	49.17	36.67	52.19	32.97
	IV	94.01	79.69	56.07	32.60	26.56	48.23	22.31
	V	94.07	77.91	52.22	25.75	19.25	42.42	17.63
S1.1-32B	I	89.84	90.40	90.00	55.83	45.42	61.62	25.45
	II	70.71	65.13	48.75	36.87	29.79	55.30	25.09
	III	61.04	54.92	35.00	25.97	19.86	58.08	25.09
	IV	56.81	47.91	26.96	20.10	14.69	52.15	24.37
	V	55.82	47.44	18.33	14.75	12.17	52.63	23.30
OpenThinker2-32B	I	96.44	96.20	95.00	68.33	52.50	62.12	56.27
	II	95.48	92.80	94.58	61.46	50.62	59.09	45.34
	III	95.27	90.48	80.25	61.94	41.53	56.73	39.19
	IV	95.19	88.63	78.04	50.21	33.96	58.46	35.39
	V	94.74	88.49	71.11	38.42	26.67	56.87	30.18
SimpleRL-Zoo-Qwen-32B	I	96.06	83.20	67.50	27.20	16.67	46.46	26.52
	II	95.40	79.80	61.67	16.67	8.12	44.19	25.81
	III	94.35	79.52	57.00	16.67	7.92	48.48	25.33
	IV	92.79	79.37	56.07	19.17	8.75	45.58	22.49
	V	91.43	76.89	53.33	14.67	10.67	46.57	22.15
Open-Reasoner-Zero-32B	I	95.83	92.00	83.75	46.67	36.67	60.10	35.13
	II	93.86	87.13	80.00	38.33	26.67	51.77	14.16
	III	92.86	83.28	68.50	35.00	26.11	50.51	13.50
	IV	91.21	80.06	68.04	27.92	22.08	47.60	11.92
	V	89.27	81.13	63.61	25.33	19.67	48.38	12.47
Qwen2.5-32B-Inst	I	95.53	82.20	60.00	20.00	16.67	42.93	26.88
	II	94.36	77.07	60.00	10.00	11.67	39.39	25.45
	III	94.17	74.76	50.00	11.11	6.39	39.06	24.25
	IV	93.37	72.20	48.04	8.33	4.17	41.79	23.30
	V	93.19	69.51	40.83	9.00	4.67	39.90	23.01

Table 22: Detailed accuracy statistics for each stress level of various LRMs.

Model \ Bench	Stress Degree	GSM8K	MATH500	AMC23	AIME24	AIME25	GPQA Diamond	LiveCode Bench
<i>API-based LRMs</i>								
DeepSeek-R1	I	96.20	97.00	93.75	81.66	68.75	70.20	63.44
	II	96.10	94.20	88.75	66.67	55.00	70.45	50.54
	III	96.20	93.40	84.75	57.78	40.00	64.48	44.44
	IV	96.18	91.94	84.11	54.17	31.67	62.88	34.59
	V	95.75	88.80	69.58	31.33	22.00	60.71	33.76
O3-Mini	I	95.83	95.00	90.00	79.16	71.66	71.21	60.21
	II	95.17	93.87	70.75	49.00	58.33	68.43	52.87
	III	94.35	80.60	59.58	40.56	38.33	67.51	51.73
	IV	93.69	88.94	58.75	30.83	27.50	66.54	47.67
	V	92.17	90.31	56.25	30.83	13.75	67.07	41.15
O4-Mini	I	93.71	90.00	96.25	73.33	80.00	76.26	70.61
	II	93.25	85.67	90.42	62.50	52.50	74.24	67.20
	III	92.96	83.40	80.75	53.89	53.33	73.40	62.96
	IV	92.97	82.83	84.82	52.50	49.17	71.97	61.56
	V	93.11	80.96	75.83	42.67	33.67	72.83	60.57
Gemini-2.5-Flash-Thinking	I	89.23	97.20	97.50	76.67	71.67	78.79	61.65
	II	89.87	87.67	75.00	65.83	47.50	72.73	53.05
	III	91.76	71.96	54.00	38.89	27.22	67.34	50.42
	IV	92.12	69.60	41.25	22.92	18.75	68.18	47.94
	V	91.35	68.20	41.39	18.00	14.33	63.74	41.94

Table 23: Detailed accuracy statistics for each stress level of various LRMs.