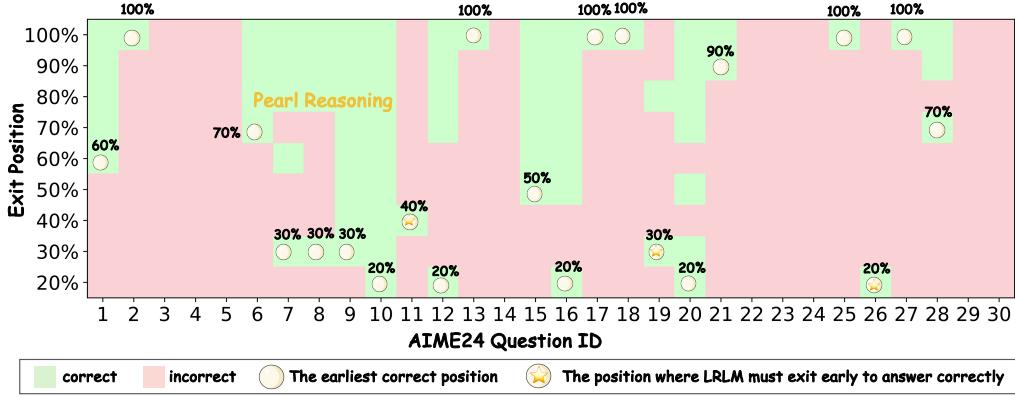


000 DYNAMIC EARLY EXIT IN REASONING MODELS

001 **Anonymous authors**

002 Paper under double-blind review



021 Figure 1: Correctness statistics for early exits at various reasoning steps.

022 ABSTRACT

023
024 Recent advances in large reasoning language models (LRMs) rely on test-time
025 scaling, which extends long chain-of-thought (CoT) generation to solve complex
026 tasks. However, overthinking in long CoT not only slows down the efficiency of
027 problem solving, but also risks accuracy loss due to the extremely detailed or re-
028 dundant reasoning steps. We propose a simple yet effective method that allows
029 LLMs to self-truncate CoT sequences by early exit during generation. Instead of
030 relying on fixed heuristics, the proposed method monitors model behavior at po-
031 tential reasoning transition points and dynamically terminates the next reasoning
032 chain's generation when the model exhibits high confidence in a trial answer. Our
033 method requires no additional training and can be seamlessly integrated into ex-
034 isting o1-like reasoning LLMs. Experiments on 10 reasoning benchmarks (e.g.,
035 GSM8K, MATH-500, AMC, GPQA, AIME and LiveCodeBench) show that the
036 proposed method is consistently effective on 11 cutting-edge reasoning LLMs of
037 varying series and sizes, reducing the length of CoT sequences by an average of
038 19.1% to 80.1% while improving accuracy by 0.3% to 5.0%.

039 1 INTRODUCTION

040 The emergence of large reasoning models (Xu et al., 2025a), such as DeepSeek-R1 (DeepSeek-AI
041 et al., 2025) and GPT-O1 (OpenAI, 2025), has marked a significant breakthrough in natural language
042 processing, particularly in solving complex and intricate tasks (WANG et al., 2025). These models
043 leverage the test-time scaling (Snell et al., 2024) law by generating a longer CoT (Wei et al., 2023)
044 with rich and diverse reasoning paths, unleashing the potential of their reasoning ability.

045 However, the generation of overlong CoT significantly increases computational overload and re-
046 reasoning latency, which hinders their deployment in computationally sensitive applications. Moreover,
047 recent research (Chen et al., 2025b; Team et al., 2025a) reveals an intrinsic overthinking problem in
048 LRM: These models persistently generate verbose reasoning sequences (Wu et al., 2025; Cuadron
049 et al., 2025), introducing irrelevant information and unnecessary thought steps. Such redundant
050 processing not only wastes computational resources but also leads to accuracy degradation by de-
051 railing from correct reasoning paths to erroneous ones (see Questions 11, 19 and 26 in Fig. 1. This
052 redundancy can be attributed to the design of the Supervised Fine-Tuning (Achiam et al., 2023;

Wei et al., 2021; Ouyang et al., 2022) or Reinforcement Learning (Bai et al., 2022; Ouyang et al., 2022; Schulman et al., 2017; Ramesh et al., 2024) stage, where the ability to dynamically adjust its reasoning length during generation is overlooked, leaving a gap in the inference efficiency of LRM.

Intuitively, as the number of reasoning paths increases, more information is referenced when generating conclusions. If we can identify the critical point where the reasoning information becomes just sufficient (termed **Pearl Reasoning**) and force the model to stop further thinking and directly output conclusions at this point, we can achieve both accuracy and efficiency. This paper aims to *find such pearls in long CoT sequences*. To validate our motivation, we forced the model to switch from thinking to directly generating answers, at different transition points in the thought process. If the answers obtained are correct, the existence of such pearl reasoning is verified. As shown in Fig. 1, about 75% samples contain such pearls (early exit yields correct answers), even 36.7% samples required only less than half of the original reasoning paths to reach correct conclusions. Therefore, how to find the pearl reasoning is a valuable topic to achieve efficient reasoning.

To this end, we propose a novel, training-free approach **DEER** that allows large reasoning language models to achieve **Dynamic Early Exit in Reasoning**. It regards the key moments when the model switches thought chains in reasoning as chances of early exit, and prompting LRM to stop thinking and generate trial answers at these moments. The confidence of each trial answer is the decision-making reference of early exit in reasoning. Specifically, the proposed method contains three actions: 1) **Reasoning Transition Monitoring**. During the generation of long CoTs, DEER monitors the positions of reasoning transitions through either linguistic marker-based (such as "Wait") or entropy-based methods. When the reasoning transition points are found, the action of 2) **Trial Answer Inducing** is triggered: we replace it with "*final answer*" tokens to induce the model to immediately generate a trial answer, which will be used for 3) **Confidence Evaluating**. If the confidence is sufficiently high, set the model to stop further thinking and generate a conclusion based on the generated thoughts. Otherwise, the action of Trial Answer Inducing is revoked, and the model continues reasoning along the original path. Moreover, Considering the potential sensitivity of models to answer inducing prompts, we propose **DEER-Pro** (a Parallel and Robust variant of DEER), which performs multiple parallel answer inductions at potential early-exit points and calibrates confidence based on the aggregated results, thereby further ensuring DEER's robustness.

Our method is simple yet effective, and can be seamlessly extended to eleven reasoning models of varying architectures and sizes, achieving excellent results in the ten reasoning benchmarks, including mathematical tasks (e.g., AIME 2024, AMC 2023 and MATH-500), scientific tasks (e.g., GPQA Diamond) and programming tasks (e.g., BigCodeBench). Specifically, our method, when integrated into cutting-edge reasoning models, can reduce the length of CoT sequences by an average of 19.1% to 80.1% while improving accuracy by 0.3% to 5.0% across different reasoning benchmarks. Our DEER offers a plug-and-play solution for improving both the efficiency and accuracy of LRM.

2 MOTIVATIONS AND OBSERVATIONS

In this section, we analyze the overthinking phenomenon in LRM and investigate the impact of static early exits on model performance. We define "pearl reasoning" as the critical juncture where reasoning information becomes precisely sufficient for accurate problem-solving. Our analysis in Figure 1 reveals that approximately 75% of samples contain such pearls (where early exit yields correct answers). Furthermore, we identified a subset of samples for which correct answers are exclusively obtainable through early exits (exemplified by Questions 11, 19, and 26 in Figure 1). Quantitative analysis presented in Figure 2(a) further demonstrates that 60.8% and 35.1% of correctly answered samples in MATH-500 and GPQA, respectively, maintain their accuracy when employing early exits after completing merely 20% of the reasoning steps. These empirical findings substantiate our hypothesis that LRM possess the potential to achieve simultaneous improvements in both computational efficiency and prediction accuracy through strategic early termination.

Fig. 2(b) illustrates that exiting at different positions corrects varying proportions of wrong answers. For the MATH dataset, the highest correction rate is achieved when exiting at 40% of the reasoning steps, whereas for the GPQA dataset, the optimal correction occurs when exiting at 50%. The optimal early exit point varies for each problem and is closely related to the inherent difficulty of the problem itself. Therefore, it is intuitive that relying on a static early exit strategy based on fixed heuristics is suboptimal, underscoring the necessity of designing a dynamic early exit mechanism.

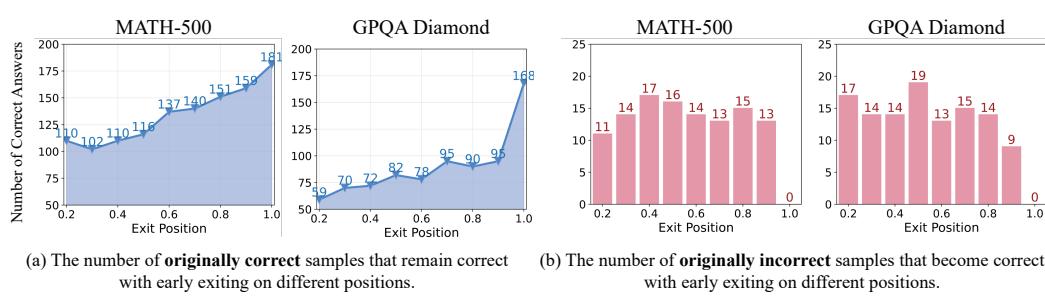


Figure 2: Quantitative pilot experiment results. Please refer to Appendix A for setups.

3 METHOD

3.1 THE GENERATION PATTERN OF LARGE REASONING MODELS

In contrast to traditional large language models (*System 1*), large reasoning models (*System 2*) (Li et al., 2025b) exhibit distinct generation patterns during the inference stage. (1) LRM_s use delimiters to divide the output into two processes: slow thinking and conclusion. LRM_s conduct systematic and thorough reasoning in the slow thinking, ultimately summarizing the thought process and providing the final answer in the conclusion. (2) During the slow thinking process, LRM_s engage in complex thinking actions (thoughts), such as problem comprehension, approach exploration, and result verification (Luo et al., 2025b). Within each reasoning action (thought), the model performs specific procedural action execution, while transitions between different reasoning actions are typically marked by action transition points (**ATP**), such as "Wait", "Alternatively".

$$\text{System 1: } [\text{Prompt}] + [\text{Completion}], \quad (1)$$

$$\text{System 2: } [\text{Prompt}] + \langle \text{think} \rangle + [\text{Slow Thinking}] + \langle / \text{think} \rangle + [\text{Conclusion}], \quad (2)$$

$$[\text{Slow Thinking}] : [\text{Action Execution}] + (\text{ATP}) + [\text{Action Execution}] + (\text{ATP}) + \dots, \quad (3)$$

where $\langle \text{think} \rangle$ and $\langle / \text{think} \rangle$ are begin-of-thinking and end-of-thinking delimiters respectively.

3.2 DYNAMIC EARLY EXIT IN REASONING

In this section, we introduce the Dynamic Early Exit in Reasoning (DEER) method to determine optimal positions for early exits (pearl reasoning path), thereby alleviating the overthinking issue.

The core idea behind DEER is that a model's confidence in its trial answer dynamically indicates whether the thinking information required for LRM_s to generate the final answer is sufficient. We observe that when the model's reasoning process is incomplete or flawed, the trial answer tends to exhibit significantly lower confidence. Conversely, when the reasoning is comprehensive and logically sound, the model generates answers with higher confidence, as illustrated in Fig. 17. This suggests that the model implicitly recognizes when **pearl reasoning** occurs, but lacks an explicit mechanism during inference to leverage this awareness for early termination. DEER aims to bridge this gap by converting implicit awareness into explicit early-exit decisions.

As shown in Fig. 3, DEER involves three designs to determine whether to exit early: reasoning transition monitor, answer inducer, and confidence evaluator.

Reasoning transition monitor. Within the DEER framework, we propose two alternative monitor design strategies: (i) **linguistic marker-based**, and (ii) **entropy-based** monitoring. For the first strategy, as mentioned in Section 3.1, LRM_s explicitly utilize ATPs to mark boundaries between different thoughts. This feature enables DEER to recognize ATPs as potential early-exit opportunities. In the second strategy, DEER employs " $\backslash n \n$ " as delimiters to demarcate reasoning steps. Following each reasoning step, DEER computes the entropy of the initial token, denoted as $H(p(\cdot | x_{<t}))$. Low entropy values indicate that the model is engaged in procedural action execution, characterized by stable reasoning patterns. Conversely, high entropy values suggest that the model is deliberating on its subsequent reasoning action, with multiple potential pathways being activated concurrently. These positions exhibiting high entropy are identified as candidate points for early-exit.

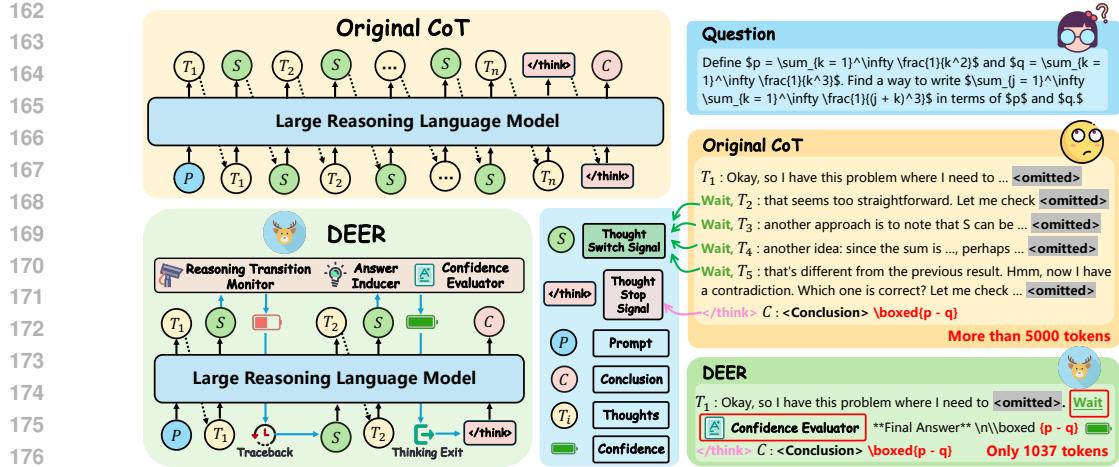


Figure 3: An overview of the Dynamic Early Exit in Reasoning (DEER) method.

Our subsequent experiments in Section 4.3 demonstrate that DEER with external linguistic markers satisfies similar properties as the second internal state-based approach while achieving comparable performance. When applied to English LRM, existing models consistently exhibit a pattern of generating such linguistic markers. Following the Occam’s Razor principle (Rasmussen & Ghahramani, 2000), we recommend adopting the first strategy. For non-English reasoning scenarios, the alternative second strategy can also accurately capture early-exit points, demonstrating the generality and robustness of DEER.

Answer inducer. When the LRM pauses at a potential early exit point, the trial answer inducer module prompts the model to generate an intermediate answer based on the reasoning content produced so far. We incorporated the answer delimiters ($\boxed{\cdot}$) into the prompt to facilitate a more precise identification of the trial answers, as follows: $A = \text{LRM}(\mathbf{P}, \mathbf{T}, \mathbf{I})$ where \mathbf{P} denotes the input prompt, \mathbf{T} denotes the generated thoughts, \mathbf{I} denotes the answer inducer prompt, and $A = [a_0, a_1, \dots, a_n]$ is the trial answer.

Confidence evaluator. The confidence evaluator computes the confidence of the induced trial answer. It takes the maximum predicted probability of each token as its confidence. For multi-token trial answers, the overall confidence is computed as their mean score across all tokens as follows:

$$p(a_t) = \text{softmax}(\mathcal{M}(\mathbf{P}, \mathbf{T}, \mathbf{I}, a_{<t})), \quad \mathcal{C} = \left(\prod_{i=1}^n \max_{a_t \in \mathcal{V}} p(a_t) \right)^{1/n} \quad (4)$$

where \mathcal{M} is the language model head at the final layer of the LRM. The calculation of \mathcal{C} employs the geometric mean, which better aligns with the multiplicative nature of joint probabilities and exhibits greater sensitivity to low probability values, thereby providing enhanced robustness.

Finally, the comparison between the obtained confidence and the empirical threshold λ determines whether to exit early. If $\mathcal{C} > \lambda$, we consider the reasoning information currently generated by the LRM to be sufficient, indicating that the model has reached the **pearl reasoning**. At this point, DEER stops further reasoning actions and proceeds to deliver the conclusion. Otherwise, the model reverts to the previous transition point to generate the next thoughts.

DEER-PRo. To further improve the reliability and accuracy of pearl reasoning identification, we introduce **DEER-PRo**, a **Parallel** and **Robust** variant of DEER. Through answer elicitation using varied prompts at early-exit points, DEER-PRo calculates both the mean and Mean Absolute Deviation (MAD) of multiple confidence values, deriving a calibrated confidence score \mathcal{C}_{cali} as follows:

$$\mathcal{C}_{cali} = \mathcal{C}_{avg} - \alpha \cdot \mathcal{C}_{MAD}, \quad \mathcal{C}_{avg} = \frac{1}{N} \sum_{i=1}^N \mathcal{C}_i, \quad \mathcal{C}_{MAD} = \frac{1}{N} \sum_{i=1}^N |\mathcal{C}_i - \mathcal{C}_{avg}| \quad (5)$$

where $\mathcal{C}_i = \left(\prod_{i=1}^n \max_{a_t \in \mathcal{V}} \text{softmax}(\mathcal{M}(\mathbf{P}, \mathbf{T}, \mathbf{I}_i, a_{<t}^i)) \right)^{1/n}$ denotes the confidence score obtained using a specific answer inducing prompt \mathbf{I}_i , N denotes the number of inducing attempts, and α is a calibration factor.

216 α is the fluctuation penalty strength coefficient. The introduced conservative bias \mathcal{C}_{MAD} effectively
217 prevents erroneous early exits caused by overestimated confidence scores resulting from positive
218 noise in prompts, where the estimated confidence exceeds the true confidence. We demonstrate in
219 the Appendix B that $\mathcal{C}_{\text{cali}}$ effectively eliminates the influence of the model’s sensitivity to answer
220 inducer prompts on early-exit accuracy, thus substantially improving DEER’s robustness.
221

222 3.3 BRANCH-PARALLEL DECODING ACCELERATION *223*

224 Intuitively, the computation of the answer inducer and confidence evaluator in DEER introduces
225 additional latency during inference, particularly in code generation tasks where trial answers remain
226 lengthy. This overhead diminishes the efficiency gains achieved through substantial reduction of
227 generated CoT sequences. To address this challenge, we integrate DEER with a branch-parallel
228 acceleration strategy (Fig. 9) that mitigates these efficiency limitations through two key mecha-
229 nisms: (1) linearization of multiple branches into a single sequence for parallel generation using
230 a specialized causal attention mask, and (2) dynamic KV cache management via confidence-based
231 pruning. This strategy facilitates temporal overlap between trial answer evaluation and concurrent
232 reasoning-chain generation, thereby optimizing overall inference efficiency.
233

234 4 EXPERIMENTS

235 4.1 EXPERIMENTAL SETUP

236 **Benchmarks, Metrics and Implementations.** We evaluate model performance across 10 bench-
237 marks, including 6 mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH-500
238 (Hendrycks et al., 2021), AMC 2023 (AI-MO, 2024), AIME 2024, AIME 2025 (MAA Committees),
239 OlympiadBench (He et al., 2024), one scientific reasoning benchmark: GPQA Diamond (Rein et al.,
240 2023), and 3 code reasoning benchmarks: HumanEval (Chen et al., 2021), BigCodeBench (Zhuo
241 et al., 2024), and LiveCodeBench (Jain et al., 2024). Among the six mathematical reasoning bench-
242 marks, GSM8K, MATH-500, and AMC 2023 are generally considered to be relatively simple rea-
243 soning tasks, whereas AIME 2024, AIME 2025, and OlympiadBench are regarded as more chal-
244 lenging. Given the extensive set of evaluation benchmarks, we selectively present the most popular
245 ones (GSM8K, MATH-500, AMC 2023, AIME 2024 and GPQA Diamond) in the main experiment.
246 More experimental results are provided in the Appendix K. We selected *Accuracy (Acc)*, *Token*
247 *Number (Tok)*, and *Compression Rate (CR)* as the evaluation metrics. **Acc** denotes the final answer
248 accuracy. **Tok** denotes the average generation length per sample to evaluate the cost. **CR** is defined
249 as the ratio of the average response length to that of the original model, with lower values indicating
250 higher compression. Given the limited number of samples in datasets AMC 2023, AIME 2024, and
251 AIME 2025, we conduct 4 sampling rounds per instance and average the results across all metrics
252 to ensure stability and reliability. We have implemented DEER using both HuggingFace Transfor-
253 mers (Wolf et al., 2020) and the vLLM inference acceleration framework (Kwon et al., 2023). The
254 experimental results presented in this paper are based on the vLLM implementation. We set the
255 hyperparameter λ to 0.95 ($\lambda = 0.95$). For entropy-based DEER, following the 80/20 principle pro-
256 posed in (Wang et al., 2025b), we designate reasoning step termination positions with entropy values
257 exceeding 0.672 as early-exit points. For DEER-Pro, we set $N = 4$ and $\alpha = 1$. More experimental
258 setup details are placed in Appendix C.
259

260 **Backbone LRM and Baselines.** We conducted experiments on the open-source DeepSeek-R1-
261 Distill-Qwen series of models (1.5B, 7B, 14B, and 32B)(DeepSeek-AI et al., 2025), Qwen3 series of
262 models (1.7B, 4B, 8B, 14B, 32B) (Qwen et al., 2025), QwQ-32B (Team, 2025), and DeepSeek-R1
263 (Liu et al., 2025c). Due to the large number of models evaluated, we selectively present DeepSeek-
264 R1-Distill-Qwen-7B, Qwen3-14B, and QwQ-32B as representative examples in the main experi-
265 ment. More experimental results are provided in the Appendix C. We compare DEER against exist-
266 ing prompt-based and output-based efficient reasoning approaches, including *Vanilla*, *TCC* (Muen-
267 nighoff et al., 2025), *CoD* (Xu et al., 2025c), *NoThinking* (Ma et al., 2025a), *Dynasor-CoT* (Fu
268 et al., 2025), and *SEAL* (Chen et al., 2025a). *Vanilla* performs direct evaluation of the LRM with-
269 out any intervention. Token-Conditional Control (*TCC*) specifies a fixed token count in the system
270 prompt to enforce a token budget; in our experiments, we set this limit based on the actual token
271 length generated by DEER. Chain-of-Draft (*CoD*) reduce verbosity by limiting the number of words
272

270 Table 1: Experimental results across various types of reasoning models. "Acc" denotes accuracy,
 271 "Tok" denotes token count, and "CR" denotes compression rate. \uparrow indicates that higher values are
 272 better, while \downarrow indicates that lower values are better. The best results are highlighted in **bold**.

274 Method	MATH												SCIENCE						Overall	
	GSM8K			MATH-500			AMC23			AIME24			GPQA-D							
	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	CR \downarrow			
DeepSeek-R1-Distill-Qwen-7B																				
<i>Vanilla</i>	89.6	1,484	100%	87.4	3,858	100%	78.8	6,792	100%	41.7	13,765	100%	23.7	10,247	100%	64.2	100%			
<i>TCC</i>	88.0	892	60.1%	89.2	3,864	100.2%	82.5	6,491	95.6%	48.4	10,603	77.0%	27.3	8,442	82.4%	67.1	83.0%			
<i>CoD</i>	84.7	298	20.1%	83.2	1,987	51.5%	77.5	4,440	65.4%	40.0	10,519	76.4%	37.9	6,431	62.8%	64.7	55.3%			
<i>NoThinking</i>	87.1	284	19.1%	80.6	834	21.6%	65.0	1,911	28.1%	26.7	4,427	32.2%	29.8	724	7.1%	57.8	21.6%			
<i>Dynasor-CoT</i>	89.6	1,285	86.6%	89.0	2,971	77.0%	85.0	5,980	88.0%	46.7	12,695	92.2%	30.5	7,639	74.5%	68.2	83.7%			
<i>SEAL</i>	88.4	811	54.6%	89.4	2,661	69.0%	—	—	—	—	—	—	—	—	—	—	—			
<i>DEER</i>	90.6	917	61.8%	89.8	2,143	55.5%	85.0	4,451	65.5%	49.2	9,839	71.5%	31.3	5,469	53.4%	69.2	61.5%			
DEER-Pro	91.0	989	66.7%	90.2	2,391	62.0%	87.5	4,877	71.8%	49.2	10,046	73.0%	30.6	5,682	55.5%	69.7	65.8%			
Qwen3-14B																				
<i>Vanilla</i>	95.1	2,047	100%	93.8	4,508	100%	95.0	7,139	100%	70.0	10,859	100%	60.1	7,339	100%	82.8	100%			
<i>TCC</i>	95.7	1,241	60.6%	94.6	4,484	99.5%	95.0	7,261	101.7%	70.8	11,573	106.6%	60.1	7,138	97.3%	83.3	93.1%			
<i>CoD</i>	85.7	648	31.7%	75.2	2,359	52.3%	72.5	4,122	57.7%	60.0	10,768	99.2%	51.0	1,177	16.0%	68.9	51.4%			
<i>NoThinking</i>	94.8	286	14.0%	85.0	1,228	27.2%	77.5	2,133	29.9%	26.7	7,337	67.6%	50.5	2,320	31.6%	66.9	34.1%			
<i>Dynasor-CoT</i>	95.6	1,483	72.4%	93.8	4,063	90.1%	95.6	6,582	92.2%	73.3	10,369	95.5%	59.6	5,968	81.3%	83.6	86.3%			
<i>DEER</i>	95.3	840	41.0%	94.0	3,074	68.2%	95.0	4,763	66.7%	76.7	7,619	70.2%	57.6	2,898	39.5%	83.7	57.1%			
DEER-Pro	95.3	926	45.2%	94.4	3,260	72.3%	95.6	4,905	68.7%	75.0	8,135	74.9%	61.2	4,062	55.4%	84.3	63.3%			
QwQ-32B																				
<i>Vanilla</i>	96.7	1,427	100%	93.8	4,508	100%	92.5	6,792	100%	66.7	10,821	100%	63.1	7,320	100%	82.6	100%			
<i>TCC</i>	95.8	1,348	94.5%	94.4	4,315	95.7%	90.0	6,818	100.4%	60.0	11,263	104.1%	61.6	7,593	103.7%	80.4	99.7%			
<i>CoD</i>	96.0	627	43.9%	94.0	3,630	80.5%	92.5	5,943	87.5%	60.0	10,731	99.2%	62.6	6,039	82.5%	81.0	78.7%			
<i>NoThinking</i>	96.2	1,113	78.0%	94.8	3,930	87.2%	87.5	6,908	101.7%	66.7	10,859	100.4%	63.6	7,668	104.8%	81.8	94.4%			
<i>Dynasor-CoT</i>	95.2	1,095	76.7%	94.2	4,176	92.6%	93.8	6,544	96.3%	63.3	11,156	103.1%	64.1	7,024	96.0%	82.1	93.0%			
<i>DEER</i>	96.3	977	68.5%	94.6	3,316	73.6%	95.0	5,782	85.1%	70.0	10,097	93.3%	64.1	6,163	84.2%	84.0	80.9%			
DEER-Pro	96.2	1032	72.3%	94.8	3,650	80.9%	95.0	5,811	85.6%	70.0	10,264	94.9%	64.7	6,201	84.7%	84.1	83.7%			

293 used in each reasoning step, focusing only on the essential calculations or transformations needed
 294 to progress. *NoThinking* prompts the model to skip the reasoning phase and directly generate the
 295 final answer. *Dynasor-CoT* periodically prompts the model to produce intermediate answers at fixed
 296 token intervals and triggers early exit when three consecutive answers are consistent. *SEAL* trains a
 297 steering vector to calibrate the CoT process, guiding the model toward more reliable reasoning.

4.2 MAIN RESULTS

300 **Overall Performance.** Due to space constraints, Tab.1 presents five widely adopted reasoning
 301 benchmarks, evaluated across three state-of-the-art reasoning models specifically covering three
 302 model scales, which comprehensively demonstrates DEER’s superior performance. We also pro-
 303 vide more results across 10 datasets covering 11 models ranging from 1.5B to 671B parameters
 304 in the Appendix. It can be found that DEER demonstrates strong adaptability across various rea-
 305 soning models and tasks, achieving accuracy improvements of 0.9 to 4.8 points while reducing se-
 306 quence length by 19.1% to 42.9% compared to vanilla models. DEER-Pro achieves higher accuracy
 307 with only a marginal increase in generation length ranging from 2.8% to 6.2% compared to DEER.
 308 We conducted comparative experiments between DEER and DEER-Pro on additional smaller-scale
 309 models. The experimental results in Table 2 demonstrate that DEER-Pro achieves more significant
 310 accuracy improvements. It indicates that DEER-Pro effectively addresses the prompt sensitivity
 311 issues in smaller models, demonstrating its superior robustness.

312 **Comparison with Efficient Reasoning SoTAs.** Tab. 1 presents comparisons between DEER and
 313 recent efficient reasoning methods. It can be observed that DEER consistently outperforms all base-
 314 lines, whereas baselines either struggle to generalize across tasks and base models, or must trade
 315 off accuracy for efficiency. Specifically, while TCC Muennighoff et al. (2025) achieve reasonable
 316 efficiency-accuracy tradeoffs on simpler tasks like GSM8K by incorporating token budgets into
 317 prompts, it fails on complex problems (such as AIME24) where models ignore prompts’ length con-
 318 straints and generate even longer responses than vanilla CoT. As for *NoThinking* and *CoD*, while
 319 achieving dramatic length reduction, they severely compromises models’ inherent reasoning capa-
 320 bilities. In contrast, Dynasor-CoT preserves reasoning quality but suffers from late termination due
 321 to its conservative early-exit condition, resulting in minimal length reduction. Notably, nearly all
 322 baselines fail completely on QwQ-32B due to the sporadic invalidation of its end-of-thinking delimiter
 323 `</think>` where the model continues generating reasoning steps after it and often produces
 324 duplicate `</think>` tokens (as shown in Appendix Fig. 18). Remarkably, DEER still achieves a
 325 19.1% length reduction on QwQ-32B despite these challenges, further demonstrating its robustness.

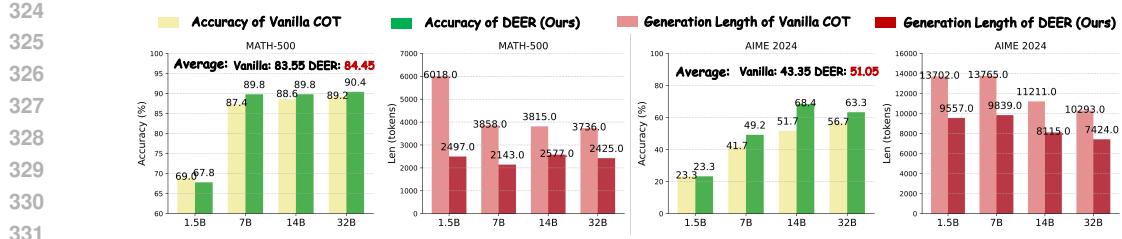


Figure 4: Experimental results of DEER compared to Vanilla CoT across DeepSeek-R1-Distill-Qwen-Series models of varying sizes on MATH-500 and AIME 2024.

Performance on Programming Tasks. Tab. 3 reports DEER’s evaluation results across three programming tasks, completing our comprehensive coverage of reasoning models’ three primary domains: mathematics, science, and programming. It demonstrates DEER’s consistent effectiveness across varying programming tasks and model sizes, achieving smaller compression ratios compared to math and science tasks (average 19.9% vs. 61.5%). This enhanced compression likely originates from the inherent characteristics of code generation, where each reasoning step typically produces verbose code segments containing substantial redundant tokens.

Performance Trends across Model Sizes and Reasoning Difficulty. Fig. 4, 10 presents evaluation results on MATH-500 and AIME 2024 datasets to examine DEER’s performance gains across different model sizes. It can be seen that DEER consistently enhances accuracy while reducing token consumption across all model sizes. A key observation is that smaller models (e.g., 1.5B) tend to generate significantly longer reasoning sequences with more severe overthinking phenomena. This stems from their limited reasoning capacity in discovering the correct reasoning steps during CoT generation. Consequently, our method achieves greater length reduction for these smaller models. Fig. 4 utilizes the MATH-500 (simple reasoning) and AIME 2024 (challenging reasoning) datasets as representative benchmarks. The results demonstrate DEER’s dual capability: it achieves more superior compression ratios on simpler problems while delivering more substantial accuracy gains on complex tasks. This precisely addresses two critical needs in reasoning systems: the efficiency demands in simple scenarios and the growing accuracy requirements in challenging scenarios.

4.3 ABLATION STUDY

Performance Trends across Token Budgets. Fig. 5 evaluates DEER’s performance across varying token budgets (controlled by different max length settings). In plots (a) and (f), the x-axis represents the actual length of model-generated CoT sequences, while the y-axis indicates model accuracy. The optimal balance between accuracy and efficiency is demonstrated by curves positioned closer to the top-left corner. The blue shaded regions quantitatively represent DEER’s performance gains: vertical height corresponds to accuracy improvement and horizontal width to token compression benefit. It can be seen that DEER consistently outperforms vanilla methods, as all points located upper-left to vanilla ones. As shown in the four-column plots on the right, we observe that vanilla models generate longer sequences with higher accuracy as token budgets increase, confirming test-time scaling. Notably, DEER demonstrates an adaptive tradeoff: under constrained token budgets, it achieves greater gains in accuracy but reduced benefits in length compression. Conversely, the opposite trend is observed with larger token budgets. This indicates that our method can dynamically adjust token budgets to meet varying requirements for accuracy-efficiency in different scenarios.

Impact of Reasoning Transition Monitor Choices. In the main experiments, we employ “Wait” as the early-exit monitoring signal, denoted as DEER(W). This simple linguistic marker-based approach yields promising results. To compare the impact of different early-exit signals on DEER performance, we conduct additional experiments using “Alternatively” as the signal, as well as entropy-based monitoring for early-exit detection. The corresponding results are presented in Tab. 5 and 6. Tab. 5 collects statistics on the number and average length of reasoning chunks obtained by dividing the original CoT with potential exit points. The chunk numbers indicate that DEER(Ent) presents the most early-exit opportunities while DEER(A) offers the fewest, exhibiting a negative correlation with average generation length. The results in Tab. 6 across additional datasets and models demonstrate that both entropy-based and linguistic marker-based monitoring exhibit comparable superior performance, significantly outperforming the baseline. In large-scale real-world deploy-

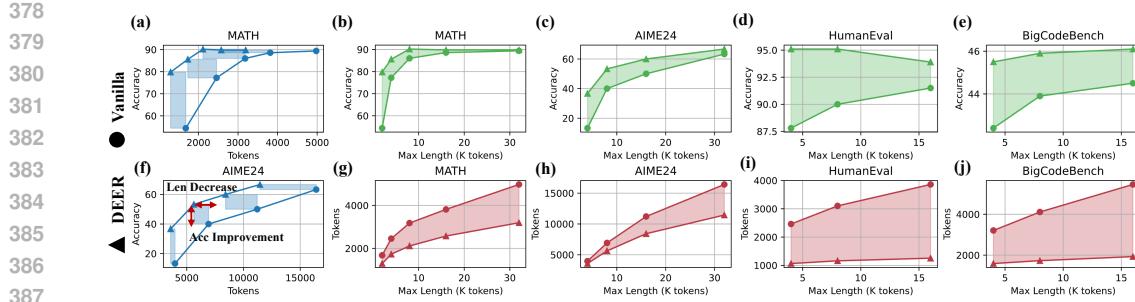


Figure 5: Performance comparison between DEER and baselines based on the DeepSeek-R1-Distill-Qwen-14B model across four datasets under different token budget settings.

ments, we advocate for the linguistic marker-based approach given its implementation simplicity and efficiency. Appendix I provides further exploration between these two monitoring strategies.

Robustness of threshold λ . Fig. 6 shows the performance of DEER on MATH-500 dataset with different threshold λ . The results indicate that when the threshold is set too low, a minor additional reduction in reasoning length leads to a significant drop in accuracy, reflecting an overcorrection of overthinking. Conversely, when the threshold is set too high, the model exits reasoning too late, resulting in prolonged reasoning lengths with a decline in accuracy. Moreover, it can be seen that our method is robust to λ within the range of 0.9-0.97, eliminating the need for hyperparameter tuning. Tab. 4 presents our robustness investigation of the threshold λ across additional datasets using Qwen3. The experimental results demonstrate that Qwen3 exhibits superior robustness to λ , maintaining consistently strong performance within the range of 0.8-0.97. Additionally, the experimental results in the appendix, conducted across 11 models and 10 datasets, uniformly employ 0.95 as the threshold value. The consistently strong results further demonstrate DEER’s generalization capability and robustness. Appendix Section J reveals that the underlying source of DEER’s robustness originates from confidence polarization phenomenon.

4.4 DISCUSSION

Efficiency Improvement. To accurately verify the gains brought by DEER and its Branch-Parallel accelerated variant in end-to-end inference efficiency, we measured the average latency on MATH-500 and AMC 2023 datasets based on huggingface transformers (Wolf et al., 2020). As shown in Fig. 7 (a), original DEER reduces the latency by 27.9% to 40.1% while the proposed branch-parallel decoding variant reduces the latency by 36.3% to 58.6%. This suggests that Branch-Parallel DEER achieves further speed improvements by efficiently reducing the latency of trial answer inducing and confidence evaluation. Additionally, we mapped the latency speedup against the length savings for every sample on MATH-500. Fig. 7 (b) illustrates that the ratio between latency speedup and length savings exhibits a superlinear trend, reinforcing the significance of DEER in enhancing inference speed. In Section H of the Appendix, we theoretically prove the efficiency of DEER and explain the underlying cause of the superlinear speedup.

Exploring the Effectiveness of DEER’s Early-Exit Mechanism. Fig. 8 presents the early-exit rate and the accuracy of early-exited samples across Qwen3-series models of varying sizes. As shown in Fig. 8(a), DEER’s early-exit rate decreases with increasing task difficulty, which accounts for its relatively lower compression performance on complex tasks compared to simpler ones. Fig. 8(b) reveals that, although early-exit accuracy declines somewhat as task difficulty increases, it remains consistently high—ranging from 88% to 98%. In a concurrent study, Zhang et al. (2025a) trained a probe to decide whether to early exit. However, their probe achieves an accuracy of only around

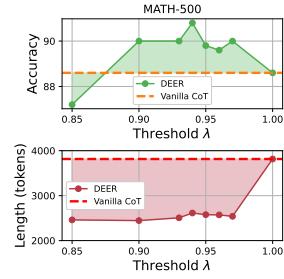


Figure 6: Impact of λ .

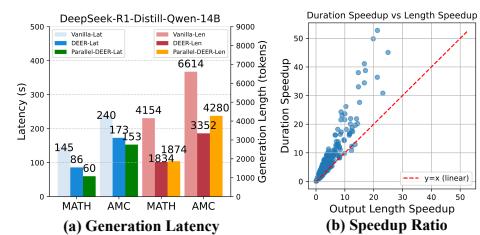


Figure 7: Efficiency Improvement.

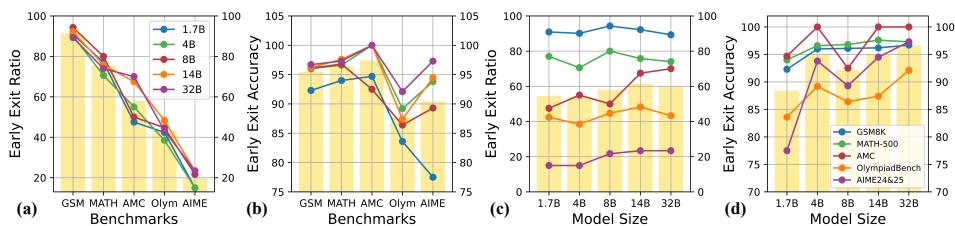


Figure 8: Early-exit rates and accuracy of early-exited samples of DEER. Figures (a) and (b) share a common legend, as do figures (c) and (d). The height of each bar reflects the average value.

80% on MATH-500, which is significantly lower than DEER’s 95%. This indicates that the model inherently possesses the ability to assess answer correctness, and that DEER effectively harnesses this capability. As illustrated in Figures 8(c) and (d), although minor differences exist across tasks of varying difficulty, there is a general trend toward higher early-exit rates and improved accuracy as model size increases. This observation implies a positive relationship between DEER’s performance and the capacity of the model: larger models yield more accurate confidence estimates, which in turn lead to better early-exit decisions. We further investigate the reasons behind DEER’s accuracy gains. Fig. 11 indicates that DEER corrects more answers (green bars) than it alters incorrectly (red bars) through early exits. This suggests that DEER not only saves computational cost by exiting early on questions it could correctly answer, but also corrects problematic thinking.

Case Study. Fig. 13 shows that both DEER and vanilla CoT arrive at the correct answer during the first reasoning step, as shown in the green box. The difference lies in the fact that DEER exits early after evaluating the confidence of the trial answer as sufficiently high, thus producing the correct result. In contrast, the vanilla CoT proceeds to the next reasoning action. After double-checking and switching reasoning approaches, the model becomes trapped in an endless cycle of verification due to inconsistent answers from the two approaches, ultimately failing to provide a final answer. Besides, Fig. 17 shows that LRM_s implicitly know when to leave early, and our method is simple and effective to realize such potential of the model itself. Please refer to Appendix L for details.

5 RELATED WORK

Following the taxonomy of efficient reasoning established in (Sui et al., 2025; Wang et al., 2025a), we categorize related work into three classes: **post-training based** methods use SFT (Yu et al., 2024; Kang et al., 2025; Xia et al., 2025; Ma et al., 2025b; Munkhbat et al., 2025; Liu et al., 2024; Han et al., 2024) with variable-length CoT data or incorporate length rewards (Team et al., 2025b; Luo et al., 2025a; Aggarwal & Welleck, 2025; Arora & Zanette, 2025; Yeo et al., 2025; Shen et al., 2025b; Qu et al., 2025; Cui et al., 2025) in reinforcement learning to enable the model to adaptively generate chains of thought of different lengths, which is beyond our training-free scope. **Prompt-based** methods (Han et al., 2024; Xu et al., 2025b; Lee et al., 2025; Renze & Guven, 2024; Chen et al., 2024) use varying prompts to enforce reasoning models to generate concise CoT with less unnecessary reasoning steps. **Output-based** methods aim to accelerate reasoning generation during the model’s decoding phase, and DEER falls into this category. However, most prior works (Xie et al., 2023; Liao et al., 2025; Li et al., 2024; Manvi et al., 2024; Aggarwal et al., 2023) focus on optimizing best-of-N sampling, which is irrelevant to our study. Instead, we select three recent concurrent works NoThinking (Ma et al., 2025a), Dynasor-CoT (Fu et al., 2025), and SEAL (Chen et al., 2025a) as baselines for comparison. More related works can be seen in Appendix M.

6 CONCLUSION

This paper verifies the rationale behind the early exit motivation in CoT generation, and accordingly proposes a training-free dynamic early exit algorithm, which makes the reasoning model withdraw from subsequent thinking when the thinking amount is just enough. Our method comprehensively evaluated across reasoning models of varying model sizes and demonstrates superior performance with fewer tokens on ten classical reasoning benchmarks, which offers a win-win solution to the trade-off between accuracy and efficiency commonly encountered in test-time scaling.

486
487

7 ETHICS STATEMENT

488
489
This work adheres to the ICLR Code of Ethics. We affirm that our research has been conducted with
490
integrity, honesty, and respect for ethical principles throughout all stages of the work.491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

- All findings presented in this paper are reported accurately and honestly. We have not fabricated, falsified, or misrepresented any data or results. Our methods and experimental procedures are described transparently to ensure reproducibility.
- All datasets used in this research were obtained and utilized in accordance with their licenses and terms of use. For any data involving personal information, we ensured compliance with privacy regulations and obtained appropriate ethical approvals where necessary.
- All contributions to this work have been properly acknowledged. We have appropriately cited all sources and prior work that influenced our research. All co-authors have made substantial contributions to the work and have agreed to the submission.
- We have carefully considered the broader implications of our work. While our research aims to advance the field positively, we acknowledge potential dual-use concerns and encourage responsible deployment of our methods in real-world applications.

504
505

8 REPRODUCIBILITY STATEMENT

506
507
To ensure the reproducibility of our results, we have provided detailed descriptions of our experimental setup, hyperparameters, and implementation details. Code and supplementary materials are
508
made available where possible to facilitate verification and future research.
509510
511

REFERENCES

512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.
Pranjal Aggarwal, Aman Madaan, Yiming Yang, et al. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. *arXiv preprint arXiv:2305.11860*, 2023.
AI-MO. Amc 2023, 2024. URL <https://huggingface.co/datasets/AI-MO/aimo-validation-amc>.
Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.
Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*, 2025.
Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec

540 Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob Mc-
 541 Grew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large
 542 language models trained on code, 2021. URL <https://arxiv.org/abs/2107.03374>.

543

544 Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. Unlocking the capa-
 545 bilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought.
 546 *Advances in Neural Information Processing Systems*, 37:54872–54904, 2024.

547

548 Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable
 549 reasoning calibration of large language models for free, 2025a. URL <https://arxiv.org/abs/2504.07986>.

550

551 Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu,
 552 Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not
 553 think that much for 2+3=? on the overthinking of o1-like llms, 2025b. URL <https://arxiv.org/abs/2412.21187>.

554

555 Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through
 556 dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

557

558 Yu-Neng Chuang, Helen Zhou, Prathusha Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki,
 559 and Xia Hu. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410*, 2024.

560

561 Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui,
 562 Vladimir Braverman, and Xia Hu. Confident or seek stronger: Exploring uncertainty-based on-
 563 device llm routing from benchmarking to generalization. *arXiv preprint arXiv:2502.04428*, 2025.

564

565 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 566 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
 567 Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.

568

569 Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu
 570 Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, et al. The danger of overthinking: Examining
 the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.

571

572 Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen
 573 Luo, Jing Huang, Zhen Li, et al. Stepwise perplexity-guided refinement for efficient chain-of-
 574 thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*, 2025.

575

576 Muzhi Dai, Chenxu Yang, and Qingyi Si. S-grpo: Early exit via reinforcement learning in reasoning
 577 models, 2025. URL <https://arxiv.org/abs/2505.07686>.

578

579 Renfei Dang, Shujian Huang, and Jiajun Chen. Internal bias in reasoning models leads to overthink-
 580 ing, 2025. URL <https://arxiv.org/abs/2505.16448>.

581

582 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,
 583 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,
 584 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao
 585 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,
 586 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
 587 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,
 588 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang
 589 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai
 590 Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang,
 591 Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang,
 592 Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang,
 593 Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang,
 R. J. Chen, R. L. Jin, Ruyi Chen, Shanghai Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng
 Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing
 Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen
 Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong

594 Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu,
 595 Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xi-
 596 aosh Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia
 597 Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng
 598 Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong
 599 Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong,
 600 Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou,
 601 Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying
 602 Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda
 603 Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu,
 604 Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu
 605 Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforce-
 606 ment learning, 2025. URL <https://arxiv.org/abs/2501.12948>.

607 Razvan-Gabriel Dumitru, Darius Peteleaza, Vikas Yadav, and Liangming Pan. Conciserl:
 608 Conciseness-guided reinforcement learning for efficient reasoning models, 2025. URL <https://arxiv.org/abs/2505.17250>.

610 Yichao Fu, Junda Chen, Yonghao Zhuang, Zheyu Fu, Ion Stoica, and Hao Zhang. Reasoning without
 611 self-doubt: More efficient chain-of-thought through certainty probing. In *ICLR 2025 Workshop*
 612 on Foundation Models in the Wild, 2025. URL <https://openreview.net/forum?id=wpK4IMJfdX>.

613 Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-
 614 budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

615 Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong
 616 Tian. Training large language models to reason in a continuous latent space. *arXiv preprint*
 617 *arXiv:2412.06769*, 2024.

618 Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi
 619 Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun.
 Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual mul-
 620 timodal scientific problems, 2024. URL <https://arxiv.org/abs/2402.14008>.

621 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
 622 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.
 623 URL <https://arxiv.org/abs/2103.03874>.

624 Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. Mit-
 625 igating overthinking in large reasoning models via manifold steering, 2025. URL <https://arxiv.org/abs/2505.22411>.

626 Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando
 627 Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free
 628 evaluation of large language models for code, 2024. URL <https://arxiv.org/abs/2403.07974>.

629 Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang,
 630 Tengchao Lv, Lei Cui, and Furu Wei. Think only when you need with large hybrid-reasoning
 631 models, 2025a. URL <https://arxiv.org/abs/2505.14631>.

632 Yuxuan Jiang, Dawei Li, and Frank Ferraro. Drp: Distilled reasoning pruning with skill-aware
 633 step decomposition for efficient large reasoning models, 2025b. URL <https://arxiv.org/abs/2505.13975>.

634 Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought
 635 without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelli-
 636 gence*, volume 39, pp. 24312–24320, 2025.

637 kvcache ai. KTransformers: A flexible framework for experiencing cutting-edge llm inference
 638 optimizations. <https://github.com/kvcache-ai/ktransformers>, 2025. URL
 639 <https://github.com/kvcache-ai/ktransformers>. GitHub repository, commit
 640 a1b2c3d, accessed 2025-05-16.

648 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.
 649 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
 650 serving with pagedattention, 2023. URL <https://arxiv.org/abs/2309.06180>.

651

652 Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought?
 653 a token complexity approach. *arXiv preprint arXiv:2503.01141*, 2025.

654

655 Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and
 656 Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv*
 657 *preprint arXiv:2401.10480*, 2024.

658

659 Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. Selfbudgeter: Adaptive
 660 token allocation for efficient llm reasoning, 2025a. URL <https://arxiv.org/abs/2505.11274>.

661

662 Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian
 663 Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang
 664 Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large
 665 language models, 2025b. URL <https://arxiv.org/abs/2502.17419>.

666

667 Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and
 668 Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint*
arXiv:2501.19324, 2025.

669

670 Hanbing Liu, Lang Cao, Yuanyi Ren, Mengyu Zhou, Haoyu Dong, Xiaojun Ma, Shi Han, and
 671 Dongmei Zhang. Bingo: Boosting efficient reasoning of llms via dynamic and significance-based
 672 reinforcement learning, 2025a. URL <https://arxiv.org/abs/2506.08125>.

673

674 Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng
 675 Zhang. Can language models learn to skip steps? *arXiv preprint arXiv:2411.01855*, 2024.

676

677 Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang,
 678 and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping, 2025b.
 679 URL <https://arxiv.org/abs/2505.15612>.

680

681 Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and
 682 Min Lin. Understanding r1-zero-like training: A critical perspective, 2025c. URL <https://arxiv.org/abs/2503.20783>.

683

684 Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao,
 685 and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning.
 686 *arXiv preprint arXiv:2501.12570*, 2025a.

687

688 Yijia Luo, Yulin Song, Xingyao Zhang, Jiaheng Liu, Weixun Wang, GengRu Chen, Wenbo Su, and
 689 Bo Zheng. Deconstructing long chain-of-thought: A structured reasoning optimization framework
 690 for long cot distillation, 2025b. URL <https://arxiv.org/abs/2503.16385>.

691

692 Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning
 693 models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025a.

694

695 Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-
 696 compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*, 2025b.

697

698 Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can
 699 predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024.

700

701 MAA Committees. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke
 Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time
 scaling, 2025. URL <https://arxiv.org/abs/2501.19393>.

702 Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-
 703 training elicits concise reasoning in large language models. *arXiv preprint arXiv:2502.20122*,
 704 2025.

705 Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez,
 706 M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024.
 707 *URL* <https://arxiv.org/abs/2406.18665>.

708 OpenAI. Learning to reason with llms. <https://openai.com/research/learning-to-reason-with-llms>, 2025. Accessed: 15 March 2025.

709 Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong
 710 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to fol-
 711 low instructions with human feedback. *Advances in neural information processing systems*, 35:
 712 27730–27744, 2022.

713 Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng,
 714 Jie Zhou, Ju Ren, and Yaoxue Zhang. Concise: Confidence-guided compression in step-by-step
 715 efficient reasoning, 2025. *URL* <https://arxiv.org/abs/2505.04881>.

716 Yuxiao Qu, Matthew YR Yang, Amritth Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan
 717 Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-
 718 tuning. *arXiv preprint arXiv:2503.07572*, 2025.

719 Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 720 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 721 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 722 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 723 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 724 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.
 725 *URL* <https://arxiv.org/abs/2412.15115>.

726 Shyam Sundhar Ramesh, Yifan Hu, Iason Chaimalas, Viraj Mehta, Pier Giuseppe Sessa,
 727 Haitham Bou Ammar, and Ilija Bogunovic. Group robust preference optimization in reward-
 728 free RLHF. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*,
 729 2024. *URL* <https://openreview.net/forum?id=PRAsjrmXXX>.

730 Carl Rasmussen and Zoubin Ghahramani. Occam's razor. In T. Leen, T. Dietterich, and
 731 V. Tresp (eds.), *Advances in Neural Information Processing Systems*, volume 13. MIT Press,
 732 2000. *URL* https://proceedings.neurips.cc/paper_files/paper/2000/file/0950ca92a4dcf426067cf2246bb5ff3-Paper.pdf.

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

736 Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in
 737 large language models. In *2024 2nd International Conference on Foundation and Large Language
 738 Models (FLLM)*, pp. 476–483. IEEE, 2024.

739 Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning
 740 with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*,
 741 2025.

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

744 Xuan Shen, Yizhou Wang, Xiangxi Shi, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. Efficient reasoning
 745 with hidden thinking. *arXiv preprint arXiv:2501.19201*, 2025a.

746 Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai
 747 Wang, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models.
 748 *arXiv preprint arXiv:2503.04472*, 2025b.

756 Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing
 757 chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*,
 758 2025c.

759

760 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally
 761 can be more effective than scaling model parameters, 2024. URL <https://arxiv.org/abs/2408.03314>.

763

764 Jiwon Song, Dongwon Jo, Yulhwa Kim, and Jae-Joon Kim. Reasoning path compression: Com-
 765 pressing generation trajectories for efficient llm reasoning, 2025. URL <https://arxiv.org/abs/2505.13866>.

766

767 DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token
 768 assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint*
 769 *arXiv:2502.03275*, 2025.

770

771 Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu,
 772 Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on effi-
 773 cient reasoning for large language models, 2025. URL <https://arxiv.org/abs/2503.16419>.

774

775 Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. Think silently, think
 776 fast: Dynamic latent compression of llm reasoning chains, 2025. URL <https://arxiv.org/abs/2505.16552>.

777

778 Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun
 779 Xiao, Chenzhuang Du, Chonghua Liao, Chunling Tang, Congcong Wang, Dehao Zhang, Enming
 780 Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han
 781 Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze
 782 Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su,
 783 Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye,
 784 Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei
 785 Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin
 786 Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia,
 787 Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang
 788 Hu, Yangyang Liu, Yanru Chen, Yeqie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping
 789 Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng
 790 Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang.
 791 Kimi k1.5: Scaling reinforcement learning with llms, 2025a. URL <https://arxiv.org/abs/2501.12599>.

792

793 Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun
 794 Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with
 795 llms. *arXiv preprint arXiv:2501.12599*, 2025b.

796

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

799

800 Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin
 801 Zhao. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage rl,
 802 2025. URL <https://arxiv.org/abs/2505.10832>.

803

804 Hongru WANG, Deng Cai, Wanjun Zhong, Shijue Huang, Jeff Z. Pan, Zeming Liu, and Kam-Fai
 805 Wong. Self-reasoning language models: Unfold hidden reasoning chains with few reasoning
 806 catalyst. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL
 807 <https://openreview.net/forum?id=p4wXiD8FX1>.

808

809 Rui Wang, Hongru Wang, Boyang Xue, Jianhui Pang, Shudong Liu, Yi Chen, Jiahao Qiu, Derek Fai
 Wong, Heng Ji, and Kam-Fai Wong. Harnessing the reasoning economy: A survey of efficient rea-
 810 soning for large language models, 2025a. URL <https://arxiv.org/abs/2503.24377>.

810 Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen,
 811 Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen
 812 Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive
 813 effective reinforcement learning for llm reasoning, 2025b. URL <https://arxiv.org/abs/2506.01939>.

814

815 Yibo Wang, Li Shen, Huanjin Yao, Tiansheng Huang, Rui Liu, Naiqiang Tan, Jiaxing Huang, Kai
 816 Zhang, and Dacheng Tao. R1-compress: Long chain-of-thought compression via chunk compres-
 817 sion and search, 2025c. URL <https://arxiv.org/abs/2505.16838>.

818

819 Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du,
 820 Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint*
 821 *arXiv:2109.01652*, 2021.

822

823 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc
 824 Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models,
 825 2023. URL <https://arxiv.org/abs/2201.11903>.

826

827 Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi,
 828 Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick
 829 von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger,
 830 Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface’s transformers: State-of-
 831 the-art natural language processing, 2020. URL <https://arxiv.org/abs/1910.03771>.

832

833 Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Under-
 834 standing chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025.

835

836 Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable
 837 chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

838

839 Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael
 840 Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Process-
 841 ing Systems*, 36:41618–41650, 2023.

842

843 Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong
 844 Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang,
 845 Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large
 846 reasoning models: A survey of reinforced reasoning with large language models, 2025a. URL
 847 <https://arxiv.org/abs/2501.09686>.

848

849 Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing
 850 less. *arXiv preprint arXiv:2502.18600*, 2025b.

851

852 Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing
 853 less, 2025c. URL <https://arxiv.org/abs/2502.18600>.

854

855 Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient
 856 reasoning with llms. *arXiv preprint arXiv:2502.12134*, 2025d.

857

858 Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more
 859 for reasoning, 2025. URL <https://arxiv.org/abs/2502.03387>.

860

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

863

864 Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen.
 865 Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large
 866 language models, 2025. URL <https://arxiv.org/abs/2505.03469>.

867

868 Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint*
 869 *arXiv:2407.06023*, 2024.

864 Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning
 865 models know when they're right: Probing hidden states for self-verification. *arXiv preprint*
 866 *arXiv:2504.05419*, 2025a.

867 Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adapthink: Reasoning models can
 868 learn when to think, 2025b. URL <https://arxiv.org/abs/2505.13417>.

870 Wenyuan Zhang, Shuaiyi Nie, Xinghua Zhang, Zefeng Zhang, and Tingwen Liu. S1-bench:
 871 A simple benchmark for evaluating system 1 thinking capability of large reasoning mod-
 872 els. *ArXiv*, abs/2504.10368, 2025c. URL <https://api.semanticscholar.org/>
 873 CorpusID:277781494.

874 Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen,
 875 and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous
 876 concept space, 2025d. URL <https://arxiv.org/abs/2505.15778>.

877 Rongzhi Zhu, Yi Liu, Zequn Sun, Yiwei Wang, and Wei Hu. When can large reasoning models
 878 save thinking? mechanistic analysis of behavioral divergence in reasoning, 2025. URL <https://arxiv.org/abs/2505.15276>.

879 Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari,
 880 Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Bench-
 881 marking code generation with diverse function calls and complex instructions. *arXiv preprint*
 882 *arXiv:2406.15877*, 2024.

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918 **A PILOT EXPERIMENT SETUP**
 919

920 We selected AIME2024 (MAA Committees) as the test set for exploratory experiments to perform
 921 a qualitative analysis and further conducted a quantitative analysis through experiments on MATH-
 922 500 (Hendrycks et al., 2021), GPQA-Diamond (Rein et al., 2023). All experiments were conducted
 923 on DeepSeek-R1-Distill-Qwen-14B (DeepSeek-AI et al., 2025). In our experiments, we first en-
 924 abled the LRM to perform a complete inference on the test set (including both the slow thinking
 925 and conclusion contents). Then, we preserved the thinking content and divided it into thinking
 926 chunks based on the action transition points. Samples with more than five thinking chunks were
 927 retained. For these samples, we retained varying proportions (20%-90%) of their thinking chunks
 928 and appended an end-of-thinking token delimiter to each truncated reasoning sequence to forcibly
 929 terminate the slow-thinking process. The model then generated its final conclusion based on the par-
 930 tial reasoning contents. For the conclusions obtained with varying thinking contents, we evaluated
 931 their correctness and presented the results of each sample in Figure 1. Furthermore, we investigated
 932 the number of samples that remained correct after early exiting when they were originally correct,
 933 as well as the number of samples that became correct after early exiting when they were originally
 934 incorrect, across three datasets in Figure 2.

935 **B PROOF OF DEER-PRO’S EFFECTIVENESS AGAINST NOISE.**
 936

937 **B.1 NOISE INDEPENDENCE OF THE MAD-CALIBRATED STRATEGY**
 938

939 Let us define the true confidence μ as the model’s actual probability of deriving the correct answer
 940 given the current reasoning content, corresponding to the real probability of pearl reasoning existing
 941 at this location. The early-exit decision threshold is denoted as λ , and the model executes early
 942 exit when $\mu > \lambda$. Since answer inducing prompts may introduce ε , the model’s output confidence
 943 fluctuates around the true confidence, yielding an observed confidence of $\mathcal{C}_i = \mu + \varepsilon_i$. In practical
 944 testing environments, we compare \mathcal{C}_i with λ to determine whether to perform early exit. Without loss
 945 of generality, we assume the noise terms ε_i to be independently and identically distributed (i.i.d.), a
 946 Gaussian distribution with mean 0 and standard deviation σ , i.e., $\varepsilon_i \sim N(0, \sigma^2)$. Here, σ represents
 947 the model’s sensitivity to prompt phrasing. A larger σ indicates higher model sensitivity, resulting
 948 in greater confidence fluctuations.

949 Next, we demonstrate DEER-PRO’s effectiveness against noise interference by comparing the de-
 950 cision error rates of DEER-PRO (\mathcal{C}_{cali}), DEER (\mathcal{C}_i), an averaging approach (\mathcal{C}_{avg}) in critical risk
 951 scenarios. Suppose \mathcal{C}_{cali} and \mathcal{C}_{avg} each conduct N times answer inducing in parallel, with N identi-
 952 cal to that in Equation (5). Given that preserving accuracy takes precedence over early-exit speedup
 953 gains in reasoning scenarios, we designate risk scenarios as those where true confidence $\mu < \lambda$. We
 954 then compute the false positive probability of observing $\mathcal{C} > \lambda$ due to noise interference.

955 **B.1.1 PROBABILITY OF ERROR FOR A SINGLE PROMPT:**
 956

$$P_{FP}(\text{Single}) = P(\mathcal{C}_i > \lambda) = P(\mu + \varepsilon_i > \lambda) = P(\varepsilon_i > \lambda - \mu) \quad (6)$$

957 Since $\varepsilon_i \sim N(0, \sigma^2)$, we can standardize it as:

$$P_{FP}(\text{Single}) = P\left(\frac{\varepsilon_i}{\sigma} > \frac{\lambda - \mu}{\sigma}\right) \quad (7)$$

958 Let $Z = \frac{\varepsilon_i}{\sigma}$, then $Z \sim N(0, 1)$, then:

$$P_{FP}(\text{Single}) = P\left(Z > \frac{\lambda - \mu}{\sigma}\right) = 1 - \Phi\left(\frac{\lambda - \mu}{\sigma}\right) \quad (8)$$

959 where Φ is the cumulative distribution function (CDF) of the standard normal distribution.

960 **B.1.2 PROBABILITY OF ERROR FOR AVERAGED CONFIDENCE:**
 961

962 For the averaged confidence \mathcal{C}_{avg} , the noise term is still Gaussian, $\varepsilon_{avg} \sim N(0, \sigma^2/N)$. The proba-
 963 bility of error for averaged confidence is:

$$P_{FP}(\text{Avg}) = P(\mathcal{C}_{avg} > \lambda) = 1 - \Phi\left(\sqrt{N} \frac{(\lambda - \mu)}{\sigma}\right) \quad (9)$$

972 Since $\frac{(\lambda-\mu)\sqrt{N}}{\sigma} > \frac{\lambda-\mu}{\sigma}$ for $N > 1$, it follows that $P_{\text{FP}}(\text{Avg}) < P_{\text{FP}}(\text{Single})$. It indicates that simply averaging multiple observed confidence values can mitigate noise interference. Nevertheless, confidence averaging fails to address the fundamental problem, as the error rate P_{FP} remains dependent on the noise standard deviation σ , increasing monotonically as σ grows. For models with substantial intrinsic noise (large σ), the parameter inside the Φ function converges to zero, driving P_{FP} toward 0.5. This indicates that high-noise models reduce to random guessing, regardless of the decision threshold λ . Therefore, the reliability of traditional approaches is severely constrained by the model's inherent noise level σ , a factor beyond our control.

980 B.1.3 PROBABILITY OF ERROR FOR MAD-CALIBRATED CONFIDENCE (DEER-PRO):

$$982 P_{\text{FP}}(\text{calibration}) = P(\mathcal{C}_{\text{cali}} > \lambda) = P(\mathcal{C}_{\text{avg}} - \alpha \cdot \mathcal{C}_{\text{MAD}} > \lambda) \quad (10)$$

984 Substituting $\mathcal{C}_{\text{avg}} = \mu + \varepsilon_{\text{avg}}$, we obtain:

$$985 P_{\text{FP}}(\text{calibration}) = P(\mu + \varepsilon_{\text{avg}} - \alpha \cdot \mathcal{C}_{\text{MAD}} > \lambda) \quad (11)$$

987 Rearranging the terms in the equation yields:

$$988 P_{\text{FP}}(\text{calibration}) = P(\varepsilon_{\text{avg}} > \alpha \cdot \mathcal{C}_{\text{MAD}} + \lambda - \mu) \quad (12)$$

989 Next, we will discuss the robustness of DEER-PRO under two distinct scenarios.

991 Scenario 1: Approximate estimation of \mathcal{C}_{MAD}

992 Under our assumption where $\varepsilon_i \sim N(0, \sigma^2)$, we have $\varepsilon_{\text{avg}} \sim N(0, \sigma^2/N)$ and $E[\mathcal{C}_{\text{MAD}}] \approx 0.8\sigma$ (993 we will provide the proof in Section B.3). For large N , the law of large numbers allows us to 994 approximate \mathcal{C}_{MAD} as 0.8σ . Given this assumption, we have:

$$995 P_{\text{FP}}(\text{calibration}) = P(\varepsilon_{\text{avg}} > \lambda + 0.8\sigma\alpha) \quad (13)$$

997 The above equation reveals that DEER-PRO fundamentally differs by employing an adaptive threshold 998 that scales with noise:

$$999 \lambda_{\text{effective}} = \lambda + 0.8\alpha\sigma \quad (14)$$

1000 We proceed to reformulate the equation by transforming $P_{\text{FP}}(\text{calibration})$ into the cumulative 1001 distribution function (CDF) of the standard normal distribution Φ . For Equation (12), we substitute 1002 $\mathcal{C}_{\text{MAD}} = 0.8\sigma$ and obtain:

$$1003 P_{\text{FP}}(\text{calibration}) = P(\varepsilon_{\text{avg}} > 0.8\sigma\alpha + \lambda - \mu) \quad (15)$$

1005 Since $\varepsilon_{\text{avg}} \sim N(0, \sigma^2/N)$:

$$1007 P_{\text{FP}}(\text{calibration}) = 1 - \Phi\left(\frac{(\lambda - \mu + 0.8\alpha\sigma)\sqrt{N}}{\sigma}\right) \quad (16)$$

1009 Simplifying:

$$1011 P_{\text{FP}}(\text{calibration}) = 1 - \Phi\left(\sqrt{N}\left(\frac{\lambda - \mu}{\sigma} + 0.8\alpha\right)\right) \quad (17)$$

1012 When noise is minimal:

$$1014 \lambda_{\text{effective}} \rightarrow \lambda \quad (18)$$

1015 The calibrated method behaves like standard thresholding, maintaining high efficiency.

1016 When noise dominates, $\frac{\lambda - \mu}{\sigma} \rightarrow 0$. The false positive rate becomes:

$$1018 P_{\text{FP}} = 1 - \Phi(0.8\alpha\sqrt{N}) \quad (19)$$

1019 which is independent of σ . It indicates that DEER-PRO effectively prevent early exit from reducing 1020 to random guessing ($P_{\text{FP}} \rightarrow 0.5$).

1022 Scenario 2: Exact computation of \mathcal{C}_{MAD} without approximation.

1023 From $\mathcal{C}_{\text{MAD}} = \frac{1}{N} \sum_{i=1}^N |\mathcal{C}_i - \mathcal{C}_{\text{avg}}|$, we know that \mathcal{C}_{MAD} is positive, therefore dividing both sides 1024 of the equation by this term, we obtain:

$$1025 P(\varepsilon_{\text{avg}}/\mathcal{C}_{\text{MAD}} > \alpha + (\lambda - \mu)/\mathcal{C}_{\text{MAD}}) \quad (20)$$

1026 Since $\mu < \lambda$, then $\alpha + (\lambda - \mu)/\mathcal{C}_{MAD} > \alpha$, so we can obtain:
 1027

$$P(\varepsilon_{avg}/\mathcal{C}_{MAD} > \alpha + (\lambda - \mu)/\mathcal{C}_{MAD}) < P(\varepsilon_{avg}/\mathcal{C}_{MAD} > \alpha) \quad (21)$$

1029 Therefore, $P(\varepsilon_{avg}/\mathcal{C}_{MAD} > \alpha)$ is an upper bound for $P_{FP}(\text{calibration})$. For the ratio $\varepsilon_{avg}/\mathcal{C}_{MAD}$,
 1030 ε_{avg} represents the signal of the noise, while \mathcal{C}_{MAD} represents the internal disorder of the noise.
 1031 Therefore, we can define $\varepsilon_{avg}/\mathcal{C}_{MAD}$ as the Signal-to-Noise Ratio (SNR).

1032 Next, let us analyze the properties of SNR. Under our assumption where $\varepsilon_i \sim N(0, \sigma^2)$, we have
 1033 $\varepsilon_{avg} \sim N(0, \sigma^2/N)$ and $E[\mathcal{C}_{MAD}] \approx 0.8\sigma$. Since both ε_{avg} and \mathcal{C}_{MAD} are proportional to σ , we can
 1034 write SNR as:
 1035

$$\text{SNR} = (\sigma * Z_{avg}) / (\sigma * Z_{MAD}) = Z_{avg}/Z_{MAD} \quad (22)$$

1036 where $Z_{avg} \sim N(0, 1/N)$ is a standardized noise mean, and Z_{MAD} is a random variable related to
 1037 MAD/σ whose distribution does not depend on σ . Hence, the probability distribution of the SNR
 1038 is independent of the model noise standard deviation σ .
 1039

$$P_{FP}(\text{calibration}) < P(Z_{avg}/Z_{MAD} > \alpha) \quad (23)$$

1040 Therefore, $P_{FP}(\text{calibration})$ is influenced only by the number of prompts N and the signal-to-noise
 1041 ratio threshold α , where larger values of N and α lead to lower error rates.
 1042

1043 Through the transformation from an absolute threshold test to a self-normalized SNR test, our MAD
 1044 strategy effectively decouples decision-making from the model's intrinsic and uncontrollable noise
 1045 level σ . In contrast to traditional approaches that break down under high-noise conditions, our
 1046 method delivers consistent, robust performance independent of model noise levels.
 1047

1048 B.2 ANALYSIS OF MAD-CALIBRATED STRATEGY'S SUPERIOR PERFORMANCE 1049

1050 In this section, we formally prove based on the event space that the false positive probability of the
 1051 MAD strategy, $P_{FP}(\text{MAD})$ ($P_{FP}(\text{calibration})$), is significantly superior to that of the simple averaging
 1052 strategy, $P_{FP}(\text{Avg})$, and consequently also outperforms $P_{FP}(\text{Single})$.
 1053

1054 **Theorem.** The false positive probability of the MAD-calibrated strategy satisfies $P_{FP}(\text{MAD}) \leq$
 1055 $\rho \cdot P_{FP}(\text{Avg})$, where $\rho = O(\exp(-\Theta(N)))$ is an exponentially decaying factor in the number of
 1056 prompts N .
 1057

Proof. The false positive events are defined as:
 1058

$$E_{\text{Avg}} = \{\varepsilon : \varepsilon_{avg} > c\} \quad (24)$$

$$E_{\text{MAD}} = \{\varepsilon : \varepsilon_{avg} - \alpha \cdot \text{MAD} > c\} \quad (25)$$

1061 where $c = \lambda - \mu > 0$ is the threshold gap, $\text{MAD} = \frac{1}{N} \sum_{i=1}^N |\varepsilon_i - \varepsilon_{avg}|$ is the mean absolute deviation,
 1062 $\varepsilon_{avg} = \frac{1}{N} \sum_{i=1}^N \varepsilon_i$ is the average noise, and there are N independent noise terms $\varepsilon_i \sim N(0, \sigma^2)$ for
 1063 $i = 1, \dots, N$.
 1064

Since $\alpha > 0$ and $\text{MAD} \geq 0$ by definition, we have:
 1065

$$\varepsilon_{avg} - \alpha \cdot \text{MAD} \leq \varepsilon_{avg} \quad (26)$$

1066 Therefore, if $\varepsilon_{avg} - \alpha \cdot \text{MAD} > c$, then necessarily $\varepsilon_{avg} > c$. This establishes:
 1067

$$E_{\text{MAD}} \subseteq E_{\text{Avg}} \quad (27)$$

1071 Consequently:
 1072

$$P_{FP}(\text{MAD}) = P(E_{\text{MAD}}) \leq P(E_{\text{Avg}}) = P_{FP}(\text{Avg}) \quad (28)$$

1074 To quantify the improvement beyond this basic inequality, we decompose the probability using
 1075 conditional probability:
 1076

$$P_{FP}(\text{MAD}) = P(E_{\text{MAD}} \cap E_{\text{Avg}}) = P(E_{\text{MAD}} | E_{\text{Avg}}) \cdot P(E_{\text{Avg}}) \quad (29)$$

1078 Since $E_{\text{MAD}} \subseteq E_{\text{Avg}}$, we have:
 1079

$$P(E_{\text{MAD}} | E_{\text{Avg}}) = P(\varepsilon_{avg} - \alpha \cdot \text{MAD} > c | \varepsilon_{avg} > c) \quad (30)$$

1080 This can be rewritten as:
 1081

$$1082 P(E_{\text{MAD}}|E_{\text{Avg}}) = P\left(\text{MAD} < \frac{\varepsilon_{\text{avg}} - c}{\alpha} \middle| \varepsilon_{\text{avg}} > c\right) \quad (31)$$

1084 Define the improvement factor:
 1085

$$1086 \rho = P\left(\text{MAD} < \frac{\varepsilon_{\text{avg}} - c}{\alpha} \middle| \varepsilon_{\text{avg}} > c\right) \quad (32)$$

1088 Then:
 1089

$$1090 P_{\text{FP}}(\text{MAD}) = \rho \cdot P_{\text{FP}}(\text{Avg}) \quad (33)$$

1091 To evaluate ρ , we analyze the structure of noise vectors that satisfy $\varepsilon_{\text{avg}} > c$. There are two primary
 1092 patterns:
 1093

1094 **Pattern A (Coherent Pattern):** All noise terms are close to c . Formally, for some small $\delta > 0$:

$$1095 \text{Pattern A} = \{\varepsilon : |\varepsilon_i - c| < \delta \text{ for all } i = 1, \dots, N\} \quad (34)$$

1097 Under Pattern A:
 1098

- 1099 • $\varepsilon_{\text{avg}} \approx c + O(\delta/\sqrt{N})$ (by the central limit theorem)
- 1100 • $\text{MAD} \leq 2\delta$ (since all values are within 2δ of each other)

1101 **Pattern B (Outlier Pattern):** A few large outliers with remaining values near zero. For example:
 1102

- 1103 • k values with $\varepsilon_i \approx Nc/k$ (large outliers)
- 1104 • $N - k$ values with $\varepsilon_i \approx 0$

1106 Under Pattern B:
 1107

- 1108 • $\varepsilon_{\text{avg}} \approx c$
- 1109 • $\text{MAD} \approx c(1 - 1/N)$ (large due to outliers)

1111 **Probability of Pattern A:**

1112 For a single noise term to fall in $(c - \delta, c + \delta)$:

$$1114 P(|\varepsilon_i - c| < \delta) = \Phi\left(\frac{c + \delta}{\sigma}\right) - \Phi\left(\frac{c - \delta}{\sigma}\right) \quad (35)$$

1116 Using Taylor expansion for small δ :

$$1118 P(|\varepsilon_i - c| < \delta) \approx \phi\left(\frac{c}{\sigma}\right) \cdot \frac{2\delta}{\sigma} = \frac{2\delta}{\sigma\sqrt{2\pi}} \exp\left(-\frac{c^2}{2\sigma^2}\right) \quad (36)$$

1121 For all N noise terms to satisfy this condition independently:

$$1123 P(\text{Pattern A}) = \left[\frac{2\delta}{\sigma\sqrt{2\pi}} \exp\left(-\frac{c^2}{2\sigma^2}\right) \right]^N \quad (37)$$

1125 **Probability of $\varepsilon_{\text{avg}} > c$:**

1127 Since $\varepsilon_{\text{avg}} \sim N(0, \sigma^2/N)$:

$$1129 P(\varepsilon_{\text{avg}} > c) = 1 - \Phi\left(\frac{c\sqrt{N}}{\sigma}\right) \quad (38)$$

1131 Using Mill's ratio approximation for large arguments:
 1132

$$1133 P(\varepsilon_{\text{avg}} > c) \approx \frac{\sigma}{c\sqrt{2\pi N}} \exp\left(-\frac{c^2 N}{2\sigma^2}\right) \quad (39)$$

1134 The key insight is that Pattern A is the only pattern where MAD remains small enough to satisfy
 1135 $\text{MAD} < (\varepsilon_{\text{avg}} - c)/\alpha$.
 1136

1137 For Pattern B and other outlier-driven patterns, $\text{MAD} = O(c)$, while $\varepsilon_{\text{avg}} - c = O(1/\sqrt{N})$ when
 1138 conditioned on $\varepsilon_{\text{avg}} \approx c$. Thus:

$$1139 \quad \text{MAD} \gg \frac{\varepsilon_{\text{avg}} - c}{\alpha} \quad (40)$$

1141 Therefore, the improvement factor is dominated by Pattern A:
 1142

$$1143 \quad \rho \lesssim \frac{P(\text{Pattern A})}{P(\varepsilon_{\text{avg}} > c)} \quad (41)$$

1146 Substituting the expressions:
 1147

$$1148 \quad \rho \lesssim \frac{\left[\frac{2\delta}{\sigma\sqrt{2\pi}} \exp\left(-\frac{c^2}{2\sigma^2}\right) \right]^N}{\frac{\sigma}{c\sqrt{2\pi N}} \exp\left(-\frac{c^2 N}{2\sigma^2}\right)} \quad (42)$$

1151 Simplifying:
 1152

$$1153 \quad \rho \lesssim \frac{c\sqrt{N}}{\sigma} \left(\frac{2\delta}{\sigma\sqrt{2\pi}} \right)^N \exp\left(-\frac{c^2 N}{2\sigma^2} + \frac{c^2 N}{2\sigma^2}\right) \quad (43)$$

$$1156 \quad \rho \lesssim c\sqrt{N} \left(\frac{2\delta}{\sigma^2\sqrt{2\pi}} \right)^N \quad (44)$$

1159 For $\delta = O(\sigma)$, let $\delta = k\sigma$ where k is a constant. Then:
 1160

$$1161 \quad \rho \lesssim c\sqrt{N} \left(\frac{2k}{\sqrt{2\pi}} \right)^N \quad (45)$$

1164 When $k < \sqrt{\pi/2}$, the term $(2k/\sqrt{2\pi})^N$ decays exponentially. The polynomial factor \sqrt{N} is
 1165 dominated by the exponential decay, yielding:
 1166

$$1167 \quad \rho = O(\sqrt{N} \cdot \exp(-\beta N)) = O(\exp(-\Theta(N))) \quad (46)$$

1169 for some positive constant β .
 1170

1171 We have established that:

$$1172 \quad P_{\text{FP}}(\text{MAD}) = \rho \cdot P_{\text{FP}}(\text{Avg}) \quad (47)$$

1174 where $\rho = O(\exp(-\Theta(N)))$ decays exponentially with the number of prompts N .
 1175

1176 This demonstrates that the MAD-calibrated strategy provides an exponential improvement over the
 1177 simple averaging approach. \square

1178 Conclusion: The MAD penalty term effectively **filters out the more probable outlier patterns**
 1179 **while only allowing the exponentially rare coherent patterns to trigger false positives**, thus
 1180 achieving superior robustness against prompt-induced noise.

1181 B.3 PROOF OF THE EXPECTED VALUE OF MAD

1183 B.3.1 THEOREM 1

1185 For a random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, the expected value of the \mathcal{C}_{MAD} is:
 1186

$$1187 \quad \mathbb{E}[\mathcal{C}_{\text{MAD}}] = \sigma \sqrt{\frac{2}{\pi}} \approx 0.8\sigma \quad (48)$$

1188 B.3.2 PROOF
11891190 **Definition.** The Mean Absolute Deviation of a random variable X with mean μ is defined as:

1191
$$\mathcal{C}_{\text{MAD}} = \mathbb{E}[|X - \mu|] \quad (49)$$

1192

1193 Let $X \sim \mathcal{N}(\mu, \sigma^2)$. Consider the standardized random variable:

1194
$$Z = \frac{X - \mu}{\sigma} \sim \mathcal{N}(0, 1) \quad (50)$$

1195

1196 Therefore:

1197
$$|X - \mu| = \sigma|Z| \quad (51)$$

1198

1199 Taking expectations on both sides:

1200
$$\mathbb{E}[|X - \mu|] = \sigma \cdot \mathbb{E}[|Z|] \quad (52)$$

1201

1200 For $Z \sim \mathcal{N}(0, 1)$ with probability density function $\phi(z) = \frac{1}{\sqrt{2\pi}}e^{-z^2/2}$:

1202
$$\mathbb{E}[|Z|] = \int_{-\infty}^{\infty} |z| \cdot \phi(z) dz \quad (53)$$

1203

1204 Due to the symmetry of the standard normal distribution about zero and the even nature of $|z|$:

1205
$$\mathbb{E}[|Z|] = 2 \int_0^{\infty} z \cdot \phi(z) dz = \frac{2}{\sqrt{2\pi}} \int_0^{\infty} z \cdot e^{-z^2/2} dz \quad (54)$$

1206

1207 Let $u = z^2/2$, then $du = z dz$. When $z = 0, u = 0$; when $z \rightarrow \infty, u \rightarrow \infty$.

1208
$$\int_0^{\infty} z \cdot e^{-z^2/2} dz = \int_0^{\infty} e^{-u} du = [-e^{-u}]_0^{\infty} = 1 \quad (55)$$

1209

1210 Substituting back:

1211
$$\mathbb{E}[|Z|] = \frac{2}{\sqrt{2\pi}} \cdot 1 = \sqrt{\frac{2}{\pi}} \quad (56)$$

1212

1213 Therefore:

1214
$$\mathcal{C}_{\text{MAD}} = \mathbb{E}[|X - \mu|] = \sigma \cdot \mathbb{E}[|Z|] = \sigma \sqrt{\frac{2}{\pi}} \approx 0.8\sigma \quad (57)$$

1215

1216 C MORE EXPERIMENT SETUP
12171218 **Metrics.** The goal of DEER is to maintain the correctness performance of LRM_s while avoiding
1219 the redundant token overhead caused by overthinking. To this end, we selected *Accuracy* (ACC)
1220 and *Generation Length* (LEN) as the evaluation metrics. *Accuracy* (ACC) is calculated as follows:
1221
$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}\{\mathcal{M}(\mathcal{LRM}(x_i)) = y_i\}$$
, where x_i is the question and y_i is the ground-truth
1222 answer from the dataset. $\mathcal{M}(\cdot)$ extracts the answer from the LRM_s’s response. $\mathbb{I}\{\cdot\}$ is an indicator
1223 function that determines whether the inside given condition is valid. The accuracy evaluation is
1224 based on the evaluation framework publicly released by Ye et al. (2025) (LIMO). Intuitively, the
1225 longer the generated text, the greater the inference cost for LRM_s. Therefore, we calculate the
1226 average generation tokens per sample to evaluate the cost as follows: $\text{Generation Length}(LEN) =$
1227
$$\frac{1}{N} \sum_{i=1}^N |\mathcal{LRM}(x_i)|$$
, where $|\cdot|$ measures the number of generated tokens. For the two programming
1228 benchmarks, we use the Pass@1 metric to measure generated code correctness.
12291230 **Implementation details.** All evaluations are conducted in a Zero-shot Chain-of-Thought (CoT)
1231 setting with the following prompt: “*Please reason step by step, and put your final answer within
1232 \boxed{\cdot}*.” For the decoding strategy, we employ greedy decoding with a single sample for the
1233 correctness evaluation. The ground-truth answers to the evaluation problems in our experiments are
1234 all well-structured numerical values or options. Therefore, we apply rule-based evaluations directly
1235 to verify mathematical equivalence. We set the maximum generation length at 16,384 to ensure
1236 that the evaluation captures complete problem-solving attempts. For DEER, the answer-inducing
1237 prompt employed is: ‘*\n\nFinal Answer\n\nboxed*’ For DEER-Pro, we additionally incorporated
1238 the following three prompts: ‘*\n\nFinal Answer\n\nBased on the analysis above, the answer is
1239 \boxed*’, ‘*\n\nFinal Answer\n\nThe correct final answer is \boxed*’, ‘*\n\nBased on the previous
1240 thinking, I believe I already know the answer.\nFinal Answer\n\nboxed*’.

1242

Algorithm 1 Dynamic Early Exit in Reasoning (DEER)

1243

```

1: Initialization: Large Reasoning Language Model  $LRM(\cdot)$ , zero-shot-CoT  $zs\_cot$ , question,
2:   answer inducer prompt  $I$ , set of action transition points  $\mathbb{P}$ , end-of-thinking delimiter  $\langle /think \rangle$ ,
3:   maximum length  $max\_len$ , and confidence threshold  $\lambda$ .
4:    $x \leftarrow zs\_cot + \text{question}$ ,  $r \leftarrow []$ 
5:   while  $\text{len}(x) < max\_len$  do
6:      $y \leftarrow LRM(x)$ 
7:     if  $y \in \mathbb{P}$  then ▷ Generate thoughts until meets action transition points
8:        $A \leftarrow LRM(x + I)$  ▷ Prompt LRM to generate trial answer tokens
9:       Get  $\mathcal{C}$  according to Equation 4 ▷ Calculate the confidence of the trial answer
10:      if  $\mathcal{C} > \lambda$  then ▷ Exit when thinking is sufficient
11:         $x \leftarrow x + \langle /think \rangle$ ,  $r \leftarrow r + \langle /think \rangle$ 
12:      end if
13:    else
14:       $x \leftarrow x + y$ ,  $r \leftarrow r + y$ 
15:    end if
16:   end while
17:   return  $r$ 

```

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

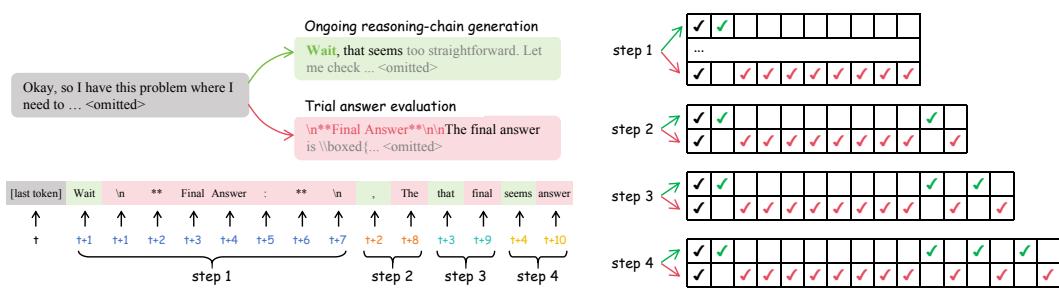


Figure 9: Branch-parallel decoding and dynamic KV cache management.

D MORE METHOD DETAILS

1274

1275

Fig. 9 illustrates the workflow of the proposed Branch-Parallel Decoding Acceleration. Algorithm 1 presents the pseudocode of DEER.

1276

E MORE BENCHMARK DESCRIPTIONS.

1277

1278

Benchmarks. To thoroughly evaluate the models’ performance across various reasoning capabilities, we have chosen 6 math reasoning benchmarks, 1 science benchmarks, and 3 coding benchmarks as follows:

1279

1280

1281

MATH BENCHMARKS:

1282

1283

- **GSM8K** is a well-curated collection of 1,319 problems in elementary mathematics. This benchmark is specifically designed to evaluate multi-step reasoning in foundational math tasks. Problems typically involve two to eight sequential operations, relying primarily on basic arithmetic performed over multiple intermediate steps.
- **MATH-500** is a challenging benchmark comprising competition-level problems drawn from diverse high school mathematics domains, including Prealgebra, Algebra, and Number Theory. For consistency with previous research, we adopt the same 500-problem subset originally curated by OpenAI for evaluation.
- **AMC 2023** contains 40 mathematical problems, covering algebra, geometry, number theory, and combinatorics. The American Mathematics Competitions (AMC), organized by the Mathematical Association of America (MAA), are prestigious contests designed to develop problem-solving skills and identify mathematical talent. For evaluation, we used 40 questions from AMC 23 in LIMO.

1296 • **AIME 2024** comprises 30 challenge problems selected from the 2024 American Invita-
 1297 tional Mathematics Examination (AIME). This prestigious contest evaluates participants'
 1298 mathematical reasoning abilities across diverse domains, including arithmetic, algebra,
 1299 counting, geometry, number theory, probability, and other secondary school math topics. A
 1300 distinctive feature of the AIME is its answer format: all solutions must be integers between
 1301 000 and 999 (inclusive). Each problem is categorized by difficulty level (1–5) according
 1302 to the Art of Problem Solving (AoPS) scale. Beyond these three math problems, we also
 1303 conducted evaluations on scientific questions.
 1304 • **AIME 2025** comprises 30 challenge problems selected from the 2025 American Invita-
 1305 tional Mathematics Examination (AIME).
 1306 • **OlympiadBench** OlympiadBench is an Olympiad-level bilingual multimodal scientific
 1307 benchmark dataset that aims to challenge and evaluate the advanced capabilities of Large
 1308 Language Models and Large Multimodal Models. It features 8,476 problems sourced from
 1309 mathematics and physics competitions at the Olympiad level, including those from the
 1310 Chinese college entrance exam. Our experimental evaluation selects the same subset of
 1311 675 samples as used in LIMO, allowing for direct rule-based evaluation of the generated
 1312 answers.

1313 **SCIENCE BENCHMARKS:**

1314 • **GPQA** is a PhD-level benchmark consisting of high-quality questions spanning physics,
 1315 chemistry, and biology subdomains. Notably, domain experts with PhDs in these fields
 1316 achieved only 69.7% accuracy on this dataset. For our experiments, we specifically select
 1317 the highest quality subset, known as **GPQA Diamond** (composed of 198 questions).
 1318

1319 **PROGRAMMING BENCHMARKS:**

1320 • **HumanEval** is proposed by OpenAI, containing 164 hand-crafted (to avoid data leakage)
 1321 Python programming tasks focusing on basic algorithms, each with function signatures,
 1322 docstrings, canonical solutions, and unit tests.
 1323 • **BigCodeBench** is designed as a real-world-oriented benchmark, which includes 1,140
 1324 tasks requiring interactions with 139 libraries and diverse function calls.
 1325 • **LiveCodeBench** is a newly proposed benchmark dataset designed to evaluate the capabili-
 1326 ties of large language models in code generation and related tasks. It aims to mitigate
 1327 issues such as test set contamination found in existing benchmarks by emphasizing scenar-
 1328 os beyond code generation, ensuring high-quality problem sources, adequate test cases,
 1329 and balanced difficulty levels. The dataset comprises problems sourced from well-known
 1330 competitive programming platforms like AtCoder, LeetCode, and CodeForces, collected
 1331 from specific time windows. Our evaluation is based on **LiveCodeBench-v5**, which con-
 1332 tains 880 programming problems collected from May 2023 to January 2025.
 1333

1334 **F COMPUTATION SOURCE**

1335 In our experiments, 8 × 80g memory H100 was used to perform evaluations.
 1336

1337 **G MORE LRM DESCRIPTIONS.**

1338 In this work, we validate the effectiveness of DEER across 12 reasoning models. The evaluated
 1339 models include: Qwen3-1.7B, Qwen3-4B, Qwen3-8B, Qwen3-14B, Qwen3-32B, DeepSeek-R1-
 1340 Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, DeepSeek-R1-Distill-Qwen-14B, DeepSeek-
 1341 R1-Distill-Qwen-32B, QwQ-32B, DeepSeek-R1-671B, and Llama-3.1-Nemotron-Nano-8B-v1. All
 1342 models in the DeepSeek-R1-Distill-Series were supervised fine-tuned using reasoning data gener-
 1343 ated by the DeepSeek-R1 model. The Qwen3-1.7B, Qwen3-4B, Qwen3-8B, and Qwen3-14B mod-
 1344 els were trained using a method known as Strong-to-Weak Distillation. Trained via reinforce-
 1345 ment learning, the non-distilled models QwQ-32B and Qwen3-32B demonstrate competitive performance
 1346 on reasoning benchmarks, matching that of DeepSeek-R1-671B. Due to computational constraints,
 1347 we implemented a quantized version of Deepseek-R1 based on KTransformers (kvcache ai, 2025).
 1348

1350 H COMPUTATIONAL COST ANALYSIS

1352 In this section, we provide a theoretical analysis to demonstrate that DEER effectively reduces computational costs. Let L denote the total length generated by the original CoT method, and α represent
 1353 DEER's compression ratio relative to L , such that DEER generates a sequence of length αL . Then,
 1354 we define k as the number of answer induction triggers within these αL tokens of reasoning and
 1355 m as the average length generated per answer induction, which is typically a small constant. Dur-
 1356 ing transformer inference, the primary computational overhead stems from attention calculations,
 1357 which constitutes our main focus. Assuming the generation process employs key-value caching
 1358 technology, each new token only needs to compute attention with the cached key-value pairs.
 1359

1360 H.1 COMPUTATIONAL COST ANALYSIS ON TIME

1361 For the original CoT method, the computational cost is:

$$1364 T = O(1) + O(2) + \dots + O(L) = O(L^2) \quad (58)$$

1366 For our DEER, The computational cost comprises two components: αL forward passes in the main
 1367 reasoning chain and km forward passes for answer inducing.
 1368

1369 First, we calculate the cost of the main reasoning chain:

$$1370 T_{\text{main}} = \sum_{t=1}^{\alpha L} t = \frac{\alpha L(\alpha L + 1)}{2} = O(\alpha^2 L^2) \quad (59)$$

1374 Next, for the computational overhead during answer inducing, we first calculate the time cost of a
 1375 single inducing. Suppose the j -th answer inducing is triggered at position p_j , yielding a cost of:
 1376

$$1377 C_{\text{single}} = \sum_{i=1}^m (p_j + i - 1) = m \cdot p_j + \sum_{i=1}^m (i - 1) = m \cdot p_j + \frac{m(m - 1)}{2} \quad (60)$$

1380 Assuming the inducing positions p_j are uniformly distributed over the interval $[0, \alpha L]$, the average
 1381 inducing position is $E[p_j] \approx \frac{\alpha L}{2}$. Hence, the total cost of answer inducing is:
 1382

$$1383 C_{\text{induce}} \approx k \cdot m \cdot \frac{\alpha L}{2} + k \cdot \frac{m(m - 1)}{2} = O(k \cdot m \cdot \alpha L) + O(k \cdot m^2) \quad (61)$$

1385 Even in the worst-case scenario where most trigger points p_j cluster near the end of reasoning, the
 1386 average inducing position is $E[p_j] \approx \alpha L$. The total cost of answer inducing is:
 1387

$$1388 C_{\text{induce}} \approx k \cdot m \cdot \alpha L + k \cdot \frac{m(m - 1)}{2} = O(k \cdot m \cdot \alpha L) + O(k \cdot m^2) \quad (62)$$

1390 As the length of each answer inducing m is negligible compared to the reasoning length L , we have:
 1391

$$1392 C_{\text{induce}} \approx O(k \cdot m \cdot \alpha L) \quad (63)$$

1393 Finally, the total cost of DEER is:

$$1394 C_{\text{DEER}} = C_{\text{main}} + C_{\text{induce}} = O(\alpha^2 L^2) + O(k \cdot m \cdot \alpha L) \quad (64)$$

1396 DEER reduces the quadratic term from $O(L^2)$ to $O(\alpha^2 L^2)$ while only introducing a linear term $O(k \cdot$
 1397 $m \cdot \alpha L)$. Since $k, m \ll L$ in long chain-of-thought reasoning, the savings from the quadratic term
 1398 reduction far exceed the overhead of the additional linear term. This analysis effectively explains
 1399 the superlinear speedup phenomenon observed in Section 4.4.
 1400

1401 H.2 COMPUTATIONAL COST ANALYSIS ON MEMORY

1402 The memory overhead analysis can be decomposed into two components: primary memory con-
 1403 sumption from the KV cache and additional overhead from parallel decoding operations.
 1404

1404
 1405 **Peak Memory Reduction.** The dominant memory overhead in modern LLM inference stems from
 1406 the storage of attention keys and values in the KV cache, whose size scales linearly with the pro-
 1407 cessed sequence length. Standard Chain-of-Thought (CoT) approaches necessitate maintaining KV
 1408 cache for all L tokens, resulting in memory complexity of $O(L)$. Through early termination at po-
 1409 sition αL where $\alpha < 1$, DEER effectively reduces the peak sequence length during inference from
 L to αL . Consequently, the peak KV cache memory consumption is reduced from $O(L)$ to $O(\alpha L)$.

1410 This memory reduction proves particularly valuable when processing long-context reasoning tasks.
 1411 Beyond reducing peak memory requirements for individual requests, this approach enables systems
 1412 to accommodate increased concurrent requests under identical memory constraints during batch
 1413 processing, thereby enhancing overall throughput.

1414
 1415
 1416 **Additional Overhead for Parallel Decoding.** The proposed parallel decoding variant, which per-
 1417 forms answer induction forward passes concurrently with main reasoning, introduces minimal addi-
 1418 tional memory overhead. This efficiency is achieved through prefix caching and sharing mechanisms
 1419 implemented in modern inference frameworks such as vLLM. When multiple reasoning branches
 1420 share a common prefix sequence, the corresponding portions of their KV caches require only single
 1421 storage in physical memory through technologies such as vLLM’s PagedAttention.

1422 During parallel decoding, the answer induction branch incurs virtually no additional KV cache over-
 1423 head, as it fully leverages the KV cache already computed by the main reasoning branch. The only
 1424 marginal additional memory requirement arises from storing a limited number of tokens represent-
 1425 ing the answer induction branch’s decoding state.

1426 For code generation tasks, our implementation incorporates specific optimizations whereby only the
 1427 initial 50 tokens are generated for confidence estimation. This design choice represents an imple-
 1428 mentation detail rather than a core methodological contribution. Experimental validation confirms
 1429 that utilizing partial answer tokens for early-exit confidence calculation remains effective for coding
 1430 tasks.

1435 I INVESTIGATION OF REASONING TRANSITION MONITORS

1436
 1437
 1438 In Section 4.3 of the main text, our experiments reveal that the choice of Reasoning Transition
 1439 Monitor exerts subtle effects on DEER, primarily manifested in how the number of potential early-
 1440 exit opportunities affects the final generation length. In this section, we investigate the underlying
 1441 connections between linguistic marker-based and entropy-based monitoring approaches.

1442 Table 10 presents a comparative analysis of average token entropy between linguistic markers and
 1443 other tokens across multiple datasets and models. Our findings reveal that linguistic markers exhibit
 1444 significantly higher entropy compared to other tokens, suggesting that the linguistic marker-based
 1445 approach inherently targets high-entropy positions where multiple candidate actions exist.

1446 Additionally, we compute cosine similarity scores between consecutive tokens in the final layer’s
 1447 hidden state representations, comparing linguistic markers with their adjacent tokens against reg-
 1448 ular token pairs. The similarity metric serves as an indicator of the model’s reasoning coherence:
 1449 high similarity reflects continuous, coherent reasoning processes, whereas low similarity signals the
 1450 occurrence of reasoning transitions. The results presented in Table 11 demonstrate that linguis-
 1451 tic markers exhibit substantially lower similarity scores, indicating disruptions in representational
 1452 continuity.

1453 Collectively, these experiments provide compelling evidence that large reasoning language mod-
 1454 els do not undergo uncertain states silently; instead, they explicitly express uncertainty through
 1455 language. The external linguistic markers leveraged by DEER constitute direct manifestations of
 1456 internal state transitions, thereby providing strong empirical support for the theoretical foundations
 1457 of our approach.

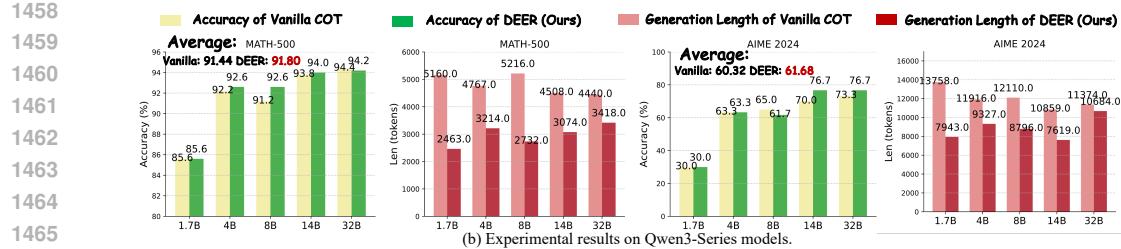


Figure 10: Experimental results of DEER compared to Vanilla CoT across Qwen3-Series models of varying sizes on MATH-500 and AIME 2024.

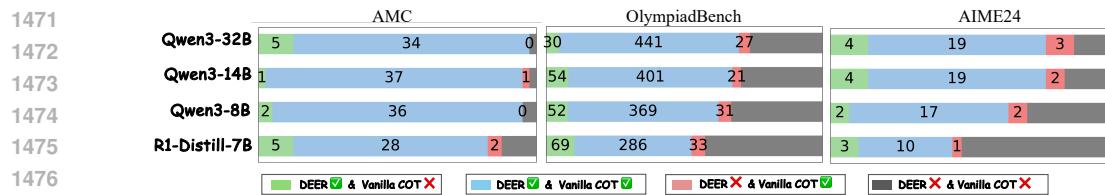


Figure 11: More detailed experimental results of DEER compared to Vanilla CoT. ✓ denotes a correct answer, and ✗ denotes an incorrect answer.

J INVESTIGATION INTO THE REASONS BEHIND DEER’S THRESHOLD ROBUSTNESS

In Section 4.3 of the main text, we demonstrate DEER’s robustness to the threshold λ through experiments across various models and datasets. In this section, we investigate the underlying source of this robustness. We analyze the confidence scores of induced answers at all potential exit positions across three models on three mathematical reasoning datasets, calculating the proportion of scores falling within three distinct intervals. Specifically, 0–0.9 represents the low-confidence interval, 0.97–1.0 represents the high-confidence interval, and 0.9–0.97 constitutes the error-prone gray zone.

The results presented in Table 12 reveal that the model’s confidence distribution exhibits a pronounced polarization phenomenon. The vast majority of cases concentrate at either the highly confident or insufficiently confident extremes, with minimal presence in the intermediate range (the error-prone gray zone). When our method induces the model to generate final answers, the confidence scores follow a distinctive U-shaped distribution, with remarkably low probability mass between 0.9 and 0.97. This phenomenon indicates that when the model possesses sufficient certainty about an answer based on its preceding reasoning chain, it generates the answer with exceptionally high probability (typically exceeding 0.99). Conversely, when uncertainty exists, the assigned probability drops substantially.

Furthermore, we observe that all three models exhibit higher proportions of high-confidence scores compared to low-confidence scores on simpler problems (GSM8K); While on more challenging problems (AIME24), the proportion of low-confidence scores exceeds that of high-confidence scores. This observation further validates the rationality of the DEER method: the confidence assigned to trial answers accurately reflects whether the existing reasoning is sufficient to solve the problem. Consequently, confidence scores are generally lower on difficult problems, leading to many failed early-exit attempts in the initial stages. This pattern also explains DEER’s varying performance across different problem difficulties. On simpler problems, the model demonstrates sufficient confidence, resulting in better compression effects. On challenging problems, the model becomes more cautious, yielding weaker compression but maintaining satisfactory accuracy. This adaptive behavior shows that DEER naturally balances computational efficiency and solution quality based on problem complexity.

1512 Table 2: Comparison of **Vanilla**, **DEER**, and **DEER-PRo** across multiple models and datasets. Acc
 1513 = accuracy (%), Len = average tokens, CR = compression ratio.

Method	GSM8K			MATH			AMC			AIME			GPQA			Overall		
	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	CR	
DeepSeek-R1-Distill-Qwen-1.5B																		
Vanilla	76.1	1,617	100%	69.0	6,018	100%	52.5	8,819	100%	23.3	13,702	100%	7.1	13,029	100%	45.6	100%	
DEER	74.7	984	60.9%	67.8	2,497	41.5%	60.0	5,496	62.3%	23.3	9,557	69.7%	12.1	5,762	44.2%	47.6	55.7%	
DEER-PRo	77.3	1,062	65.7%	70.0	2,891	48.0%	62.5	5,701	64.6%	26.7	10,390	75.8%	14.5	6,820	52.3%	50.2	61.3%	
Qwen3-4B																		
Vanilla	94.1	2,175	100%	92.2	4,767	100%	87.5	7,443	100%	63.3	11,916	100%	46.5	9,294	100%	76.7	100%	
DEER	94.5	1,250	57.5%	92.6	3,214	67.4%	87.5	4,906	65.9%	63.3	9,327	78.3%	47.5	3,275	35.2%	77.1	60.9%	
DEER-PRo	94.5	1,301	59.8%	93.0	3,517	73.8%	92.5	5,153	69.2%	65.0	9,651	81.0%	49.2	3,750	40.3%	78.8	64.8%	
Qwen3-1.7B																		
Vanilla	90.1	2,045	100%	85.6	5,160	100%	70.0	8,637	100%	30.0	13,758	100%	35.9	9,271	100%	62.3	100%	
DEER	90.3	1,066	52.1%	85.6	2,463	47.7%	70.0	4,673	54.1%	30.0	7,943	57.7%	43.4	3,549	38.3%	63.9	50.0%	
DEER-PRo	90.7	1,261	61.7%	87.2	2,702	52.4%	75.0	5,143	59.5%	35.0	8,644	62.8%	44.5	3,960	42.7%	66.5	55.8%	

1526
 1527 Table 3: Experimental results on programming tasks. Acc = accuracy (%), Tok. = average tokens,
 1528 CR = compression ratio.

Model	Method	HumanEval			BigCodeBench			LiveCodeBench			Overall	
		Acc	Tok.	CR	Acc	Tok.	CR	Acc	Tok.	CR	Acc	CR
R1-Distill-Qwen Series												
32B	Vanilla	91.5	3,861	100%	44.5	5,459	100%	56.0	9,109	100%	64.0	100%
	DEER	93.9	1,254	32.5%	46.1	1,929	35.3%	56.6	3,677	40.4%	65.5	36.1%
14B	Vanilla	89.0	4,039	100%	40.9	4,806	100%	52.7	9,259	100%	60.9	100%
	DEER	90.9	1,000	24.8%	40.7	1,583	32.9%	52.1	4,000	43.2%	61.2	33.6%
7B	Vanilla	78.6	5,666	100%	26.1	8,516	100%	38.4	10,482	100%	47.7	100%
	DEER	78.6	913	16.1%	25.2	1,605	18.8%	40.3	2,582	24.6%	48.0	19.9%
Qwen3 Series												
14B	Vanilla	93.3	3,277	100%	44.6	5,072	100%	73.4	8,203	100%	70.4	100%
	DEER	93.9	1,118	34.1%	44.3	792	15.6%	74.1	5,437	66.3%	70.8	38.7%
8B	Vanilla	85.4	3,904	100%	37.9	6,994	100%	64.7	8,871	100%	62.7	100%
	DEER	87.8	793	20.3%	41.1	608	8.7%	65.1	4,257	48.0%	64.7	25.7%
4B	Vanilla	91.5	3,768	100%	36.1	7,804	100%	64.7	8,789	100%	64.1	100%
	DEER	92.7	1,050	27.9%	37.1	826	10.6%	63.5	5,626	64.0%	64.4	34.2%
1.7B	Vanilla	83.5	3,580	100%	25.7	6,151	100%	51.7	9,447	100%	53.6	100%
	DEER	84.1	2,236	62.5%	28.3	2,104	34.2%	51.4	8,425	89.2%	54.6	62.0%

K MORE EXPERIMENTAL RESULTS

1551 **More experiments across Model Sizes on Qwen3.** The performance of the Qwen3-series models
 1552 across model sizes and reasoning difficulty in Fig. 10 is consistent with the findings presented in
 1553 Section 4.2.

Performance on SoTA Reasoning Models.

1554 We evaluated DEER’s effectiveness on two state-of-the-art reasoning models: Qwen3-32B (representing dense models) and Deepseek-R1 671B (representing MoE models). To fully leverage their reasoning capabilities, we set their maximum sequence lengths to the officially recommended 32k and 16k, respectively. The impact of max length will be further discussed in the next section. Due to computational constraints, we implemented a quantized version of Deepseek-R1 based on KTransformers (kvcache ai, 2025). Fig. 12 provides a close look at DEER’s performance on two challenging datasets, AIME and MATH.

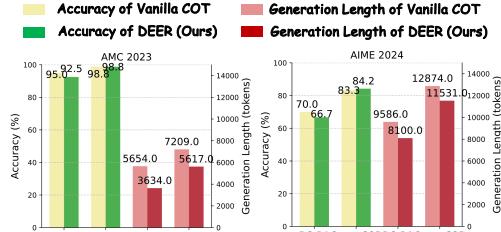


Figure 12: Performance on SoTA models.

1566 Table 4: Additional threshold sensitivity experiments across more models and tasks.
1567

	Qwen3-14B	GSM8K	MATH	AMC
1568	Vanilla	95.1	93.8	95.0
1569	0.80	96.0	93.8	93.8
1570	0.85	95.7	94.4	93.8
1571	0.90	96.1	93.8	95.0
1572	0.95	96.0	94.0	95.6
1573	0.97	95.7	93.8	94.4
1574				
	Qwen3-8B	GSM8K	MATH	AMC
1575	Vanilla	94.9	91.2	87.5
1576	0.80	95.2	91.4	88.8
1577	0.85	94.9	92.0	90.0
1578	0.90	95.5	92.8	91.3
1579	0.95	95.3	93.2	92.5
1580	0.97	95.3	93.0	92.5
1581				

1582
1583 The results show that DEER maintains competitive accuracy (with R1 making only one additional
1584 error on each dataset) while significantly reducing sequence length by 10.4% - 35.7%.

1585
1586
1587 **Performance Trends across More Model Sizes and Benchmarks.** To provide a more comprehensive
1588 demonstration of DEER’s effectiveness and facilitate comparison for researchers, we present
1589 experimental results on seven reasoning benchmarks and eleven large reasoning language models.
1590 Fig. 13 compares the experimental results between DEER and vanilla CoT, demonstrating that the
1591 conclusions drawn in the main text of the paper hold consistently across more benchmarks and
1592 additional models.

1593 In addition to popular reasoning models, we also evaluate DEER on less commonly used models. As
1594 mentioned in the Limitations section, Llama-3.1-Nemotron-Nano-8B-v1 consistently exhibits low
1595 confidence in generating intermediate answers, resulting in a significantly lower early stopping rate
1596 during evaluation compared to mainstream models (Qwen3-8B: 80%, R1-Distill-Qwen-7B: 85%,
1597 Llama-3.1-Nemotron-Nano-8B-v1: 55%). Consequently, as shown in Table 5, the improvement
1598 in reasoning efficiency for Llama-3.1-Nemotron-Nano-8B-v1 is limited. Nevertheless, DEER still
1599 effectively mitigates overthinking in this model, as evidenced by its ability to prevent subsequent
1600 reasoning steps from altering correct answers into incorrect ones through early stopping.

1601
1602 **Performance with Different Decoding Configurations.** As the configuration of DeepSeek-R1-
1603 Distill-Series models recommends a maximum length of 16k (16,384), we evaluate Qwen3-14B
1604 under the same setting in the main experiments to maintain setup consistency. In practice, this
1605 length is sufficient for most real-world applications. In addition, to ensure experimental stability
1606 and reproducibility, we employ greedy decoding in the main experiments. Nevertheless, to provide a
1607 more comprehensive assessment of DEER’s performance, we further conduct experiments on larger
1608 variants of the Qwen3-series models (8B, 14B, 32B) using the officially recommended decoding
1609 strategy (`max_len = 32768, top_p = 0.95, temperature = 0.6`). Fig. 8 shows that DEER remains
significantly effective under these configurations, demonstrating the robustness of our approach.

1610
1611 **Performance with Different Multi-Token Confidence Averaging Methods.** In the main text,
1612 we mentioned adopting the geometric mean strategy for calculating multi-token answer confidence
1613 scores, as it better aligns with the multiplicative nature of joint probability computation in language
1614 models and exhibits higher sensitivity to low probability values. In this section, we supplement our
1615 analysis with comparative experiments using arithmetic mean calculation (DEER-AM), employing
1616 the same early-exit threshold of 0.95. The results in Table 7 demonstrate that DEER-AM exhibits a
1617 significant decrease in accuracy compared to DEER-GM, while achieving marginal improvements
1618 in compression ratio. This indicates that the arithmetic mean dilutes the contribution of low-valued
1619 tokens, resulting in overall inflated confidence scores and consequently leading to premature in-
correct exits. Therefore, we recommend using the geometric mean for estimating true confidence
scores.

1620 Table 5: Results on **MATH-500** (DeepSeek-R1-Distill-Qwen-14B) with different reasoning transi-
 1621 tion monitors. DEER(W) denotes transition via *Wait*, DEER(A) via *Alternatively*, and DEER(Ent)
 1622 via entropy threshold. Chunk Size denotes the length (token numbers) of one reasoning chunk
 1623 (thought), and Chunk Num denotes the number of reasoning chunks.

Method	Accuracy	Tokens	Chunk Size	Chunk Num	Exit Ratio	Exit Acc.
Vanilla	88.6	3815	—	—	—	—
DEER(W)	89.6 _{+1.0}	2572 _{-32.6%}	259.5	14.7	87.6%	93.4%
DEER(A)	90.8 _{+2.2}	2775 _{-27.3%}	719.8	5.3	54.8%	91.2%
DEER(Ent)	90.2 _{+1.6}	2339 _{-38.7%}	183.7	20.8	90.2%	93.0%

1631
 1632 Table 6: Comparison of **Vanilla**, **DEER-W**, and **DEER-Ent** across multiple models and datasets.
 1633 Acc = accuracy (%), Len = average tokens, CR = compression ratio.

Method	GSM8K			MATH			AMC			AIME			GPQA			Overall	
	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	CR
<i>DeepSeek-R1-Distill-Qwen-7B</i>																	
Vanilla	89.6	1,484	100%	87.4	3,858	100%	78.8	6,792	100%	41.7	13,765	100%	23.7	10,247	100%	64.2	100%
DEER-W	90.6	917	61.8%	89.8	2,143	55.5%	85.0	4,451	65.5%	49.2	9,839	71.5%	31.3	5,469	53.4%	69.2	61.5%
DEER-Ent	90.8	876	59.0%	89.2	2,261	58.6%	85.0	4,072	60.0%	48.4	8,961	65.1%	29.6	5,037	49.2%	68.6	58.4%
<i>Qwen3-14B</i>																	
Vanilla	95.1	2,047	100%	93.8	4,508	100%	95.0	7,139	100%	70.0	10,859	100%	60.1	7,339	100%	82.8	100%
DEER-W	95.3	840	41.0%	94.0	3,074	68.2%	95.0	4,763	66.7%	76.7	7,619	70.2%	57.6	2,898	39.5%	83.7	57.1%
DEER-Ent	96.1	803	39.2%	93.8	2,979	66.1%	93.3	4,903	68.7%	73.3	7,128	65.6%	58.1	2,818	38.4%	82.9	55.6%
<i>Qwen3-8B</i>																	
Vanilla	94.9	2,245	100%	91.2	5,216	100%	87.5	7,986	100%	65.0	12,110	100%	51.5	9,145	100%	78.0	100%
DEER-W	95.2	1,071	47.7%	92.6	2,732	52.4%	92.5	4,392	55.0%	61.7	8,796	72.6%	52.5	3,111	34.0%	78.9	52.3%
DEER-Ent	95.8	1,037	46.2%	93.6	2,789	53.5%	91.3	4,003	50.1%	63.3	8,328	68.8%	51.5	3,248	35.5%	79.1	50.8%

1648 **Error Bars with 95% Confidence Intervals.** To demonstrate the statistical significance of
 1649 DEER’s accuracy gains, we conducted multiple experimental runs on two models and calculated
 1650 error bars with 95% confidence intervals. Specifically, we performed four independent runs on
 1651 **GSM8K**, **MATH**, and **GPQA** benchmarks. Given the limited sample sizes of **AMC23** and **AIME24**,
 1652 we increased the number of experimental repetitions to eight for these datasets. The results pre-
 1653 sented in Table 9 confirm that the accuracy improvements achieved by our method are statistically
 1654 significant.

L CASE STUDY DETAILS

1656
 1657 In Fig. 13, we provide examples of results on **MATH-500** to visually demonstrate the effectiveness
 1658 of DEER. The design of DEER ensures that it follows the same reasoning process as the vanilla CoT
 1659 method before early exiting. Both methods arrive at the correct answer during the first reasoning
 1660 step, as shown in the green box. The difference lies in the fact that our method exits early after
 1661 evaluating the confidence of the trial answer as sufficiently high, thus producing the correct result.
 1662 In contrast, the vanilla CoT method proceeds to the next reasoning action. After double-checking
 1663 and switching reasoning approaches, the model becomes trapped in an endless cycle of verification
 1664 due to inconsistent answers from the two approaches, ultimately failing to provide a final answer. In
 1665 addition, Fig. 14, 15, 16 provides additional generated examples to more comprehensively demon-
 1666 strate the effectiveness of DEER’s early-exit mechanism and illustrate the underlying mechanisms
 1667 of the approach.

1668 Figure 17 shows the detailed process of DEER applied on a mathematical example. It can be ob-
 1669 served that, at each reasoning switch point (“Wait” token), DEER generates a trial answer and
 1670 evaluates its confidence. The change in confidence is consistent with the reliability of the current
 1671 reasoning chunks and trial answers. This shows that LMRs implicitly know when to leave early, and
 1672 our method is simple and effective to realize such potential of the model itself.

1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
Table 7: Comparison of **Vanilla**, **DEER-GM** (Geometric Mean), and **DEER-AM** (Arithmetic Mean) across multiple models and datasets. Acc = accuracy (%), Len = average tokens, CR = compression ratio.

Method	GSM8K			MATH			AMC			AIME			GPQA			Overall	
	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	Len	CR	Acc	CR
DeepSeek-R1-Distill-Qwen-7B																	
Vanilla	89.6	1,484	100%	87.4	3,858	100%	78.8	6,792	100%	41.7	13,765	100%	23.7	10,247	100%	64.2	100%
DEER-GM	90.6	917	61.8%	89.8	2,143	55.5%	85.0	4,451	65.5%	49.2	9,839	71.5%	31.3	5,469	53.4%	69.2	61.5%
DEER-AM	90.2	832	56.1%	88.2	1,879	48.7%	80.0	3,872	57.0%	43.3	8,095	58.8%	22.6	4,116	40.2%	64.9	52.2%
Qwen3-14B																	
Vanilla	95.1	2,047	100%	93.8	4,508	100%	95.0	7,139	100%	70.0	10,859	100%	60.1	7,339	100%	82.8	100%
DEER-GM	95.3	840	41.0%	94.0	3,074	68.2%	95.0	4,763	66.7%	76.7	7,619	70.2%	57.6	2,898	39.5%	83.7	57.1%
DEER-AM	95.3	811	39.6%	92.4	2,620	58.1%	90.0	4,513	63.2%	63.3	6,933	63.8%	53.6	2,508	34.2%	78.9	51.8%
Qwen3-8B																	
Vanilla	94.9	2,245	100%	91.2	5,216	100%	87.5	7,986	100%	65.0	12,110	100%	51.5	9,145	100%	78.0	100%
DEER-GM	95.2	1,071	47.7%	92.6	2,732	52.4%	92.5	4,392	55.0%	61.7	8,796	72.6%	52.5	3,111	34.0%	78.9	52.3%
DEER-AM	94.9	972	43.3%	92.0	2,522	48.4%	87.5	3,899	48.8%	56.7	7,697	63.6%	49.7	2,950	32.3%	76.2	47.3%

M RELATED WORK DETAILS

The advent of Open-AI o1 (OpenAI, 2025) established test-time scaling (Snell et al., 2024) as a pivotal research direction in the LLM community. This approach enhances LLMs’ slow thinking capabilities, enabling breakthroughs in complex problem solving. The recent open-sourcing of DeepSeek-R1 (DeepSeek-AI et al., 2025) has further intensified interest in locally deployed reasoning models. However, two critical challenges have emerged: 1) excessively long CoT generated significantly degrades inference efficiency, and 2) growing empirical evidence (Chen et al., 2025b; Team et al., 2025a) reveals their susceptibility to overthinking – a phenomenon where models continue reasoning beyond the point of optimal output. Zhang et al. (2025c) introduces a novel benchmark named S1-Bench to test the performance of LRM on simple tasks, evaluating the overthinking issues of these LRM. Following the taxonomy of efficient reasoning established in (Sui et al., 2025; Wang et al., 2025a), we categorize related work into three classes: post-training based, prompt-based, and output-based efficient reasoning methods.

Post-training based efficient reasoning methods use supervised fine-tuning (Yu et al., 2024; Kang et al., 2025; Xia et al., 2025; Ma et al., 2025b; Munkhbat et al., 2025; Zhu et al., 2025; Liu et al., 2024; Han et al., 2024; Qiao et al., 2025; Yu et al., 2025) with variable-length CoT data or incorporate length rewards (Team et al., 2025b; Luo et al., 2025a; Aggarwal & Welleck, 2025; Arora & Zanette, 2025; Yeo et al., 2025; Shen et al., 2025b; Qu et al., 2025; Cui et al., 2025; Dai et al., 2025; Liu et al., 2025a,b; Tu et al., 2025; Wang et al., 2025c; Dumitru et al., 2025; Li et al., 2025a; Jiang et al., 2025a; Zhang et al., 2025b) in reinforcement learning to enable the model to adaptively generate chains of thought of different lengths. However, these methods often require a large amount of computational resources and face challenges in dataset construction. Recently, some work (Hao et al., 2024; Shen et al., 2025c; Cheng & Van Durme, 2024; Dang et al., 2025; Shen et al., 2025a; Su et al., 2025; Tan et al., 2025; Saunshi et al., 2025; Zhang et al., 2025d) has shown that using latent representations to replace explicit textual reasoning steps allows reasoning models to be more efficient. However, such methods often require extensive-epoch SFT on carefully curated datasets(Hao et al., 2024; Xu et al., 2025d), leading to overfitting on the output format and consequently compromising the model’s inherent expressiveness and generalization ability.

Prompt-based efficient reasoning methods (Han et al., 2024; Xu et al., 2025b; Lee et al., 2025; Renze & Guven, 2024; Chen et al., 2024) use varying prompts to enforce reasoning models to generate concise CoT with less unnecessary reasoning steps. Especially, (Aytes et al., 2025; Chuang et al., 2024; 2025; Ong et al.) assign different prompts to queries based on their difficulty, thereby adjusting the length of the CoT generated by reasoning models. We also explored the performance of our method combined with prompt design in Tab. 1, demonstrating further reductions in the length of reasoning chains while maintaining considerable accuracy.

Most of the **Output-based efficient reasoning** methods focus on optimizing the best-of-N sampling for LLMs, such as pruning low-quality samples (Xie et al., 2023; Liao et al., 2025) and implementing early stopping (Li et al., 2024; Manvi et al., 2024; Aggarwal et al., 2023) when multiple samples

1728
1729 Table 8: Experimental results on the Qwen3-series models under the officially recommended set-
1730 tings (max_len = 32768, top_p = 0.95, temperature = 0.6).
1731

Budget	Method	GSM8K		MATH-500		AMC23		AIME24		AIME25		OlympiadB		Overall	
		Acc	Tok.	Acc	Tok.	Acc	Tok.	Acc	Tok.	Acc	Tok.	Acc	Tok.	Acc	CR
Qwen3-8B															
32k	Vanilla	95.7	2246	93.8	5368	93.8	9424	70.0	16717	65.0	17880	67.9	11025	81.0	100%
	DEER	95.5	981	94.0	3227	95.0	5898	75.8	12465	63.3	15135	67.0	9075	81.8	68.0%
Qwen3-14B															
32k	Vanilla	95.7	1699	94.8	4800	95.0	6837	75.0	14347	76.7	16437	68.7	9992	84.3	100%
	DEER	95.8	933	95.0	3301	96.3	6299	74.2	10896	76.7	15014	68.9	8263	84.5	77.6%
Qwen3-32B															
32k	Vanilla	96.0	1714	95.8	4609	98.8	7209	83.3	12874	78.3	15292	69.3	9775	86.9	100%
	DEER	95.8	992	95.4	3325	98.8	5617	84.2	11531	78.3	13981	69.8	8671	87.1	79.6%

1741 Table 9: Accuracy performance on reasoning benchmarks with 95% confidence intervals.
1742

Model	GSM8K	MATH	AMC23	AIME24	GPQA
Vanilla (ds-7B)	0.897 [0.891, 0.902]	0.877 [0.869, 0.884]	0.794 [0.767, 0.821]	0.425 [0.400, 0.449]	0.247 [0.203, 0.291]
DEER (ds-7B)	0.904 [0.896, 0.912]	0.897 [0.883, 0.911]	0.856 [0.835, 0.878]	0.492 [0.463, 0.520]	0.299 [0.257, 0.341]
Vanilla (Qwen3-14B)	0.948 [0.942, 0.955]	0.938 [0.932, 0.943]	0.938 [0.918, 0.957]	0.708 [0.660, 0.757]	0.596 [0.571, 0.621]
DEER (Qwen3-14B)	0.955 [0.949, 0.962]	0.942 [0.936, 0.948]	0.953 [0.940, 0.967]	0.754 [0.718, 0.791]	0.587 [0.566, 0.608]

1743
1744 achieve self-consistency. However, following the introduction of advanced reasoning models like
1745 R1, there is less reliance on best-of-N sampling methods, as these models exhibit strong reasoning
1746 capabilities independently. Very recently, two concurrent works share similar motivations with
1747 ours. Zhang et al. (2025a) also proposes to terminate early based on trial answers, but requires an
1748 additional probe model to determine the correctness. They focus on enhancing the verification capa-
1749 bilities of the probe model, whereas our method explore how to enable the model to self-determine
1750 when to exit early and integrate seamlessly into existing reasoning logic. Ma et al. (2025a) prompts
1751 reasoning models to directly output final answers during decoding, but only achieves better per-
1752 formance in the low-budget regime or being adapted to best-of-N methods compared to baselines,
1753 which limits the applicability and generalization. Song et al. (2025) periodically compresses the KV
1754 cache by retaining KV cache that receive high importance score to accelerates inference by leverag-
1755 ing the semantic sparsity of reasoning paths. Jiang et al. (2025b) uses a teacher model to perform
1756 skill-aware step decomposition and content pruning, and then distills the pruned reasoning paths
1757 into a student model. Huang et al. (2025) projects the steering direction onto the low-dimensional
1758 activation manifold and intervenes the activations to reduce thinking tokens.
1759
1760

1761

N USE OF LLMs

1762

1763 In the preparation of this manuscript, Large Language Models (LLMs) were employed as auxiliary
1764 tools for Language Polishing. During the final stages of manuscript preparation, LLMs were util-
1765 ized to refine the language of selected passages, including grammar checking, sentence structure
1766 optimization, and expression standardization. This process was limited to linguistic improvements
1767 and did not involve the generation or modification of any substantive academic content, including
1768 research insights, data analysis, or conclusion derivation. It should be emphasized that all core argu-
1769 ments, research methodologies, experimental designs, data analyses, and conclusions presented in
1770 this paper were independently developed by the authors. LLMs served solely as language processing
1771 aids, and the authors assume full academic responsibility for all content.
1772
1773

1774
1775
1776
1777
1778
1779
1780
1781

1782

1783

1784

1785

Table 10: Token Entropy for *Linguistic Markers* and *Other Tokens*.

1786

1787

	Qwen3-8B	Linguistic Markers	Other Tokens
1788	gsm8k	0.901	0.438
1789	math	1.058	0.385
1790	gpqa	1.269	0.500
1791	DS-7B	Linguistic Markers	Other Tokens
1792	gsm8k	1.550	0.658
1793	math	1.753	0.565
1794	gpqa	1.241	0.510
1795			

1796

1797

1798

1799

1800

1801

1802

Table 11: Hidden States Cosine Similarity between *Linguistic Markers* and *Other Tokens*.

1803

1804

	Qwen3-8B	Linguistic Markers	Other Tokens
1805	gsm8k	0.262	0.543
1806	math	0.237	0.493
1807	gpqa	0.240	0.509
1808	DS-7B	Linguistic Markers	Other Tokens
1809	gsm8k	0.306	0.608
1810	math	0.247	0.530
1811	gpqa	0.231	0.505
1812			

1813

1814

1815

1816

1817

1818

1819

Table 12: Confidence interval distribution (%) across tasks for different models.

1820

1821

	Qwen3-14B	0–0.9	0.9–0.97	0.97–1.0
1822	gsm8k	38.92	5.95	55.06
1823	math	49.53	4.83	45.46
1824	aime	77.20	2.45	20.35
	Qwen3-8B	0–0.9	0.9–0.97	0.97–1.0
1826	gsm8k	35.31	5.34	59.12
1827	math	45.27	4.58	49.98
1828	aime	78.61	2.39	19.00
	DS-7B	0–0.9	0.9–0.97	0.97–1.0
1831	gsm8k	26.54	5.29	68.17
1832	math	34.09	5.46	60.45
1833	aime	80.90	1.41	17.69

1834

1835

1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889

Table 13: Experimental results on more types of reasoning models and reasoning benchmarks. "Acc" denotes accuracy, "Tok" denotes token count, and "CR" denotes compression rate. \uparrow indicates that higher values are better, while \downarrow indicates that lower values are better. The best results are highlighted in **bold**.

Method	GSM8K			MATH-500			AMC23			AIME24			AIME25			OlympiadBench			SCIENCE			Overall			
	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	Acc \uparrow	Tok \downarrow	CR \downarrow	GPQA-D	Tok \downarrow	CR \downarrow	Acc \uparrow	CR \downarrow		
DeepSeek-RI-Distill-Qwen-32B																									
<i>Vanilla</i>	94.3	1,202	100%	89.2	3,736	100%	87.5	5,354	100%	56.7	10,293	100%	43.3	11,075	100%	55.7	7,334	100%	56.1	7,181	100%	69.0	100%	69.0	100%
<i>DEER</i>	95.1	819	68.1%	90.4	2,425	64.9%	95	4,252	79.4%	63.3	7,424	72.1%	46.7	8,913	80.5%	57.8	5,351	73.0%	64.1	4,943	68.8%	73.2	72.4%		
DeepSeek-RI-Distill-Qwen-14B																									
<i>Vanilla</i>	93.9	1,458	100%	88.6	3,815	100%	82.5	6,545	100%	51.7	11,211	100%	36.7	12,304	100%	52.6	7,908	100%	52.0	6,731	100%	65.4	100%	65.4	100%
<i>DEER</i>	93.3	1,040	71.3%	89.8	2,577	67.5%	85.0	4,240	64.8%	68.4	8,115	72.4%	36.7	10,125	82.3%	55.0	5,736	72.5%	56.6	4,856	72.1%	69.3	71.9%		
DeepSeek-RI-Distill-Qwen-7B																									
<i>Vanilla</i>	89.6	1,484	100%	87.4	3,858	100%	78.8	6,792	100%	41.7	13,765	100%	26.7%	12,767	100%	47.3	8,563	100%	23.7	10,247	100%	56.5	100%	56.5	100%
<i>DEER</i>	90.6	917	61.8%	89.8	2,143	55.5%	85.0	4,451	65.5%	49.2	9,839	71.5%	36.7	7,257	56.8%	52.6	5,420	63.3%	31.3	5,449	53.4%	62.2	61.1%		
DeepSeek-RI-Distill-Qwen-1.5B																									
<i>Vanilla</i>	76.1	1,617	100%	69.0	6,018	100%	52.5	8,819	100%	23.3	13,702	100%	13.3	14,450	100%	28.0	11,200	100%	7.1	13,029	100%	38.5	100%	38.5	100%
<i>DEER</i>	74.7	984	60.9%	67.8	2,497	41.5%	60.0	5,496	62.3%	23.3	9,557	69.7%	10.0	9,281	64.2%	32.0	5,960	53.2%	12.1	5,762	44.2%	40.0	56.6%		
Qwen-3-32B																									
<i>Vanilla</i>	96.3	1,668	100%	94.4	4,440	100%	95.0	7,627	100%	73.3	11,374	100%	65.0	12,446	100%	63.4	6,438	100%	65.2	6,893	100%	78.9	100%	78.9	100%
<i>DEER</i>	96.2	769	46.1%	94.2	3,418	77.0%	97.5	5,753	75.4%	76.7	8,682	76.3%	66.7	10,893	87.5%	67.9	5,189	80.6%	64.7	4,167	60.5%	80.6	71.9%		
Qwen-3-14B																									
<i>Vanilla</i>	95.1	2,047	100%	93.8	4,508	100%	95.0	7,139	100%	70.0	10,859	100%	63.3	12,286	100%	62.5	8,692	100%	60.1	7,339	100%	77.1	100%	77.1	100%
<i>DEER</i>	95.3	840	41.0%	94.0	3,074	68.2%	95.0	4,763	66.7%	76.7	7,619	70.2%	66.7	11,135	90.6%	67.4	7,060	81.2%	57.6	2,898	39.5%	79.0	65.0%		
Qwen-3-8B																									
<i>Vanilla</i>	94.9	2,245	100%	91.2	5,216	100%	87.5	7,986	100%	65.0	12,110	100%	54.2	12,835	100%	59.3	9,487	100%	51.5	9,145	100%	71.9	100%	71.9	100%
<i>DEER</i>	95.2	1,071	47.7%	92.6	2,732	52.4%	92.5	4,392	55.0%	61.7	8,796	72.6%	60.0	12,229	95.3%	62.4	7,479	78.8%	52.5	3,111	34.0%	73.8	62.3%		
Qwen-3-4B																									
<i>Vanilla</i>	94.1	2,175	100%	92.2	4,767	100%	87.5	7,443	100%	63.3	11,916	100%	48.4	13,112	100%	59.3	9,098	100%	46.5	9,294	100%	70.2	100%	70.2	100%
<i>DEER</i>	94.5	1,250	57.5%	92.6	3,214	67.4%	87.5	4,906	65.9%	63.3	9,327	78.3%	55.0	12,039	91.8%	64.7	7,569	83.2%	47.5	3,275	35.4%	72.2	68.5%		
Qwen-3-17B																									
<i>Vanilla</i>	90.1	2,045	100%	85.6	5,160	100%	70.0	8,637	100%	30.0	13,758	100%	26.7	13,943	100%	52.2	9,437	100%	35.9	9,271	100%	55.8	100%	55.8	100%
<i>DEER</i>	90.3	1,066	52.1%	85.6	2,463	47.7%	70.0	4,673	54.1%	30.0	7,943	57.7%	36.7	11,579	83.0%	52.6	7,257	76.9%	43.4	3,549	38.3%	58.4	58.6%		
Qwen-32B																									
<i>Vanilla</i>	96.7	1,427	100%	93.8	4,508	100%	92.5	6,792	100%	66.7	10,821	100%	46.7	12,300	100%	65.2	8,546	100%	63.1	7,320	100%	75.0	100%	75.0	100%
<i>DEER</i>	96.3	977	68.5%	94.6	3,316	73.6%	95.0	5,782	85.1%	70.0	10,097	93.3%	50.0	11,598	94.3%	65.2	7,639	89.4%	64.1	6,163	84.2%	76.5	84.0%		
Llama-3.1-Nemotron-Nano-8Bv1																									
<i>Vanilla</i>	89.2	1,618	100%	91.2	3,794	100%	90.0	6,153	100%	56.7	10,821	100%	35.0	13,192	100%	54.3	7,321	100%	41.9	8,074	100%	65.5	100%	65.5	100%
<i>DEER</i>	89.8	1,473	91.0%	91.4	2,995	78.9%	90.0	5,408	87.9%	66.7	9,755	90.6%	36.7	11,820	89.6%	60.7	6,407	87.5%	47.5	7,576	73.8%	69.0	88.5%		

1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1939
 1940
 1941
 1942
 1943

Question: A Senate committee has 5 Democrats, 5 Republicans, and 1 Independent. In how many ways can they sit around a circular table if all the members of each party all sit next to each other? (Two seatings are considered equivalent if one is a rotation of the other.)

Answer: DEER

[Slow thinking]

Okay, so I have this problem about a Senate committee with 5 Democrats, 5 Republicans, and 1 Independent. They need to sit around a circular table, and all the members of each party have to sit next to each other. I need to figure out in how many different ways this can happen. The problem also mentions that two seatings are considered the same if one is just a rotation of the other. Hmm, okay, let me break this down.

First, since it's a circular table, the number of arrangements is usually less than a straight line because rotations are considered the same.

<omitted>

So, let me think. There are three parties: Democrats, Republicans, and Independents. Each party is a group that needs to be seated together. So, instead of thinking of 11 individuals, I can think of three blocks: D (Democrats), R (Republicans), and I (Independent). Each block has to be seated together, so the problem reduces to arranging these three blocks around the table.

<omitted>

So, for three blocks, it's $2! = 2$ distinct arrangements.

But now, within each block, the members can be arranged among themselves. For the Democrats, there are 5 people, so they can be arranged in $5!$ ways. Similarly, the Republicans can be arranged in $5!$ ways. The Independent is just one person, so there's only 1 way to arrange that block.

Therefore, the total number of arrangements is the number of ways to arrange the blocks multiplied by the number of ways to arrange the people within each block. So, that would be $2! * (5! * 5! * 1!) = 2 * (120 * 120 * 1) = 2 * 14400 = 28800.$

</think> **Early Exit!** ✓

[Conclusion]

The Senate committee consists of three parties: Democrats (D), Republicans (R), and an Independent (I). Since they must sit together, we treat each party as a single block.

1. **Arranging the Blocks**: Around a circular table, the number of ways to arrange 3 blocks is $3! = 6$.
2. **Arranging Within Blocks**: - Democrats can be arranged in $5!$ ways. - Republicans can be arranged in $5!$ ways. - The Independent is a single person, so there's $1!$ way.
3. **Total Arrangements**: Multiply the arrangements of the blocks by the arrangements within each block: $6 * 5! * 5! * 1! = 6 * 120 * 120 * 1 = 28,800$

****Answer**: \boxed{28800}**

Token number: 870

Vanilla CoT

[Slow thinking]

Okay, so I have this problem about a Senate committee with 5 Democrats, 5 Republicans, and 1 Independent. They need to sit around a circular table, and all the members of each party have to sit next to each other.

<omitted>

Therefore, the total number of arrangements is the number of ways to arrange the blocks multiplied by the number of ways to arrange the people within each block. So, that would be $2! * (5! * 5! * 1!) = 2 * (120 * 120 * 1) = 2 * 14400 = 28800.$

Wait, hold on. Is that all? Let me double-check. So, we have three blocks: D, R, I. Arrange them around the table: $(3-1)! = 2$. Then, within each block, arrange the members: $5!$ for D, $5!$ for R, and $1!$ for I. So, yes, $2 * 5! * 5! * 1! = 2 * 120 * 120 = 28800.$ ✓

But wait, another thought: since the table is circular, does the direction matter? Like, clockwise vs. counterclockwise. In circular permutations, sometimes arrangements that are mirror images are considered different unless specified otherwise.

<omitted>

But let me think again. Is there another way to approach this problem? Maybe by considering the circular arrangement as a linear arrangement where we fix one person's position to eliminate rotational symmetry.

<omitted>

But since we've fixed the Democrats, the remaining seats are in a line, so the number of ways to arrange the Republican block and the Independent is $2!$ (since there are two blocks: R and I). Then, within the Republican block, they can be arranged in $5!$ ways, and the Independent is just 1.

So, the total number of arrangements would be 1 (for the fixed Democrat) * $2! * 5! * 1! = 2 * 120 = 240.$ ✗

Wait, that's different from the previous answer. Hmm, so which one is correct? Wait, no, actually, if we fix one Democrat's seat, we're not just fixing that one seat, but the entire block of Democrats. So, the number of ways to arrange the blocks is $2!$ (for R and I), and within each block, it's $5!$ for R and $1!$ for I. So, $2! * 5! * 1! = 2 * 120 = 240.$ But that seems too low because earlier we had 28800.

Wait, so which approach is correct? Let me think.

<omitted>

Wait, so which approach is correct? Let me think.

<omitted>

Wait, so which approach is correct? Let me think.

Repetition

[No Conclusion]

Token number: 16384

Figure 13: Comparison of generated content samples between DEER and Vanilla CoT on MATH-500. Both DEER and vanilla CoT arrive at the correct answer during the first reasoning step, as shown in the green box. The difference lies in the fact that DEER exits early after evaluating the confidence of the trial answer as sufficiently high, thus producing the correct result.

1944
1945
1946
1947
1948
1949
1950
1951
1952
1953

1954 **Question:** The coordinates of a parallelogram are $(5, 3)$, $(6, 8)$, $(7, 4)$ and (x, y) and $x > 7$.
1955 What is the value of $x + y$?

1956 **Answer:** **DEER**

1957 Okay, so I have this problem where I need to find the
1958 coordinates of the fourth vertex of a parallelogram. The
1959 given points are $(5, 3)$, $(6, 8)$, $(7, 4)$, and (x, y) with the
1960 condition that $x > 7$. Then, I need to find the value of $x + y$.
1961 Hmm, let me think about how to approach this.
1962 I remember that in a parallelogram, opposite sides are equal
1963 and parallel. That means the midpoints of the diagonals
1964 should be the same. So, if I can find the midpoint of one
1965 diagonal, it should be equal to the midpoint of the other
1966 diagonal.
1967 Let me label the points to make it clearer. Let's
1968 say point A is $(5, 3)$, point B is $(6, 8)$, point C is $(7, 4)$, and
1969 point D is (x, y) . Now, I need to figure out which points are
1970 connected to each other. Since it's a parallelogram, the
1971 order of the points matters. I think the points should be
1972 connected in order, so A to B to C to D and back to A. But
1973 I'm not entirely sure, so maybe I should consider different
1974 possibilities.
1975 <omitted>

1976 Let me try another approach.
1977 Another property of a parallelogram is that the diagonals bisect each other. So, the
1978 midpoint of diagonal AC should be the same as the
1979 midpoint of diagonal BD.
1980 Let me calculate the midpoint of AC first. Point A is $(5, 3)$ and point C is $(7, 4)$. The midpoint
1981 formula is $((x_1 + x_2)/2, (y_1 + y_2)/2)$. So, midpoint of AC is $((5 + 7)/2, (3 + 4)/2) = (12/2, 7/2) = (6, 3.5)$.

1982 <omitted>
1983 Midpoint is $((5 + x)/2, (3 + y)/2)$.
1984 Midpoint of BC: Point B is $(6, 8)$ and point C is $(7, 4)$. Midpoint is $((6 + 7)/2, (8 + 4)/2) = (13/2, 12/2) = (6.5, 6)$.
1985 Setting midpoints equal:
1986 $(5 + x)/2 = 6.5$ and $(3 + y)/2 = 6$.
1987 Solving the first equation: $(5 + x)/2 = 6.5 \Rightarrow 5 + x = 13 \Rightarrow x = 8$.
1988 Solving the second equation: $(3 + y)/2 = 6 \Rightarrow 3 + y = 12 \Rightarrow y = 9$.
1989 Okay, this gives $x = 8$ and $y = 9$, which satisfies $x > 7$.
1990 So, the fourth point is $(8, 9)$, and $x + y = 8 + 9 = 17$.
1991 </think>

1992 To find the coordinates of the fourth vertex of the
1993 parallelogram, we use the property that the diagonals of a
1994 parallelogram bisect each other.
1995 <math>\text{Points B (6, 8) and C (7, 4).}1996 <math>\text{- Midpoint formula: } ((6+7)/2, (8+4)/2) = (6.5, 6)1997 <math>\text{Setting midpoints equal: } ((5+x)/2, (3+y)/2) = (6.5, 6)1998 <math>\text{Solving the first equation: } (5+x)/2 = 6.5 \Rightarrow 5+x = 13 \Rightarrow x = 81999 <math>\text{Solving the second equation: } (3+y)/2 = 6 \Rightarrow 3+y = 12 \Rightarrow y = 92000 $\text{So, the fourth point is (8, 9), and } x+y = 8+9 = 172001$
2002 <math>\text{Answer: } \boxed{17}

2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097

2098 **Token number:** ~800

2099 **Vanilla CoT**

2100 Okay, so I have this problem where I need to find the
2101 coordinates of the fourth vertex of a parallelogram. The
2102 given points are $(5, 3)$, $(6, 8)$, $(7, 4)$, and (x, y) with the
2103 condition that $x > 7$. Then, I need to find the value of $x + y$.
2104 Hmm, let me think about how to approach this.
2105 I remember that in a parallelogram, opposite sides are equal
2106 and parallel. That means the midpoints of the diagonals
2107 should be the same. So, if I can find the midpoint of one
2108 diagonal, it should be equal to the midpoint of the other
2109 diagonal.
2110 Let me label the points to make it clearer. Let's
2111 say point A is $(5, 3)$, point B is $(6, 8)$, point C is $(7, 4)$, and
2112 point D is (x, y) . Now, I need to figure out which points are
2113 connected to each other. Since it's a parallelogram, the
2114 order of the points matters. I think the points should be
2115 connected in order, so A to B to C to D and back to A. But
2116 I'm not entirely sure, so maybe I should consider different
2117 possibilities.

2118 Let me try another approach.
2119 Another property of a parallelogram is that the diagonals bisect each other. So, the
2120 midpoint of diagonal AC should be the same as the
2121 midpoint of diagonal BD.

2122 <omitted>

2123 Midpoint is $((5+x)/2, (3+y)/2)$.
2124 Midpoint of BC: Point B is $(6, 8)$ and point C is $(7, 4)$. Midpoint is $((6+7)/2, (8+4)/2) = (13/2, 12/2) = (6.5, 6)$.
2125 Setting midpoints equal:
2126 $(5+x)/2 = 6.5 \Rightarrow 5+x = 13 \Rightarrow x = 8$.
2127 Solving the second equation: $(3+y)/2 = 6 \Rightarrow 3+y = 12 \Rightarrow y = 9$.
2128 Okay, this gives $x = 8$ and $y = 9$, which satisfies $x > 7$.
2129 So, the fourth point is $(8, 9)$, and $x+y = 8+9 = 17$.

2130 **Wait, let me verify this.** If the points are $A(5,3)$, $B(6,8)$,
2131 $C(7,4)$, and $D(8,9)$, does this form a parallelogram?
2132 Let me check the vectors.

2133 <omitted>

2134 **These aren't equal either.** Hmm, that's confusing.
2135 **Wait, maybe I made a mistake** in assigning the points. Let me try
2136 another approach.

2137 <omitted>

2138 **Hmm, these aren't equal. Wait, that's a problem.**
2139 Wait, but according to the vector addition, D should be $(8,9)$. So,
2140 why are vectors AD and BC not equal?
2141 **Wait, maybe I need to check the other pair of sides.** Vector AD and vector
2142 BC are not equal, but vector AB and vector DC are equal.

2143 <omitted>

2144 **These aren't equal, but in a parallelogram, both pairs should
2145 be equal.** So, that's a problem.
2146 Wait, but according to the midpoint method, it's a parallelogram. So, maybe I'm
2147 missing something.

2148 <omitted>

2149 **No answer !**

2150 **Token number:** 16384

2151 **Figure 14:** Comparison of generated content between DEER and Vanilla on MATH-500.

1998

1999

2000

2001

2002

2003

2004

2005

2006

2007

2008

2009

2010

2011

2012

2013

2014

2015

2016

2017

2018

2019

2020

2021

2022

2023

2024

2025

2026

2027

2028

2029

2030

2031

2032

2033

2034

2035

2036

2037

2038

2039

2040

2041

2042

2043

2044

2045

2046

2047

2048

2049

2050

2051

Question: Point A lies somewhere within or on the square which has opposite corners at $(0,0)$ and $(2,2)$. Point B lies somewhere within or on the square which has opposite

Answer: corners at points $(4,2)$ and $(5,3)$. What is the greatest possible value of the slope of the line containing points A and B ? Express your answer as a common fraction.

DEER

Okay, so I have this problem where I need to find the greatest possible slope of a line connecting two points, A and B. Point A is somewhere inside or on a square with opposite corners at $(0,0)$ and $(2,2)$. Point B is inside or on another square with opposite corners at $(4,2)$ and $(5,3)$. I need to figure out the maximum slope possible between any two points A and B in these squares.
First, let me visualize the squares. The first square has corners at $(0,0)$ and $(2,2)$. Similarly, the second square has opposite corners at $(4,2)$ and $(5,3)$. Hmm, so that square is a bit smaller. Let me figure out its sides. The distance between $(4,2)$ and $(5,3)$ is $\sqrt{(5-4)^2 + (3-2)^2} = \sqrt{1+1} = \sqrt{2}$. So, it's a square with side length $\sqrt{2}$, rotated 45 degrees relative to the axes.
Wait, actually, no. If the opposite corners are $(4,2)$ and $(5,3)$, then the sides are not aligned with the axes.
<omitted>

But wait, is that achievable? Because point A is in the first square, so if A is at $(2,0)$, which is a corner, and point B is at $(4,3)$, which is a corner of the second square. So, connecting $(2,0)$ to $(4,3)$ would give a slope of $(3-0)/(4-2) = 3/2$. But is that the maximum?
</think>

To find the greatest possible slope of the line containing points A and B , we need to maximize the expression $\frac{y_B - y_A}{x_B - x_A}$. Point A lies within or on the square with opposite corners at $(0,0)$ and $(2,2)$. The minimum y -coordinate for A is 0 and the maximum y -coordinate is 2 . Point B lies within or on the square with opposite corners at $(4,2)$ and $(5,3)$. The maximum y -coordinate for B is 3 and the minimum x -coordinate is 4 . To maximize the slope, we should choose A at $(2,0)$ and B at $(4,3)$. The slope is then $\frac{3-0}{4-2} = \frac{3}{2}$. However, upon closer inspection, the square for B is actually a diamond with corners at $(4,2)$, $(5,3)$, $(5,2)$, and $(4,3)$. The point $(4,3)$ is indeed a corner of this square. Therefore, the maximum slope is achievable.
The greatest possible slope is $\frac{3}{2}$. "gold_answer": " $\frac{3}{2}$ "

Token number: ~1000

Vanilla CoT

Okay, so I have this problem where I need to find the greatest possible slope of a line connecting two points, A and B. Point A is somewhere inside or on a square with opposite corners at $(0,0)$ and $(2,2)$. Point B is inside or on another square with opposite corners at $(4,2)$ and $(5,3)$. I need to figure out the maximum slope possible between any two points A and B in these squares.
First, let me visualize the squares. The first square has corners at $(0,0)$ and $(2,2)$. Similarly, the second square has opposite corners at $(4,2)$ and $(5,3)$. Hmm, so that square is a bit smaller. Let me figure out its sides. The distance between $(4,2)$ and $(5,3)$ is $\sqrt{(5-4)^2 + (3-2)^2} = \sqrt{1+1} = \sqrt{2}$. So, it's a square with side length $\sqrt{2}$, rotated 45 degrees relative to the axes.
Wait, actually, no. If the opposite corners are $(4,2)$ and $(5,3)$, then the sides are not aligned with the axes.
<omitted>

But wait, is that achievable? Because point A is in the first square, so if A is at $(2,0)$, which is a corner, and point B is at $(4,3)$, which is a corner of the second square. So, connecting $(2,0)$ to $(4,3)$ would give a slope of $(3-0)/(4-2) = 3/2$. But is that the maximum?

Wait, but maybe we can get a steeper slope by choosing points that are not at the corners. Because if we can have a larger $y_B - y_A$ with a smaller $x_B - x_A$, that would give a steeper slope.
<omitted>

Wait, but hold on. Let me think again. Maybe if we don't take A at $(2,0)$ and B at $(4,3)$, but somewhere else, we can get a steeper slope.
<omitted>

But wait, is there a way to have a larger slope? For example, if A is at $(2,2)$ and B is at $(4,3)$. Then, the slope is $(3-2)/(4-2) = 1/2$, which is less than $3/2$. Alternatively, if A is at $(0,0)$ and B is at $(5,3)$, then the slope is $(3-0)/(5-0) = 3/5$, which is less than $3/2$.
<omitted>

Alternatively, if point A is at (x, y_A) where $x < 2$, then point B would have to be at $(x+2, y_B)$. But since the second square starts at $x=4$, $x+2$ must be at least 4, so x must be at least 2.
<omitted>

No answer !

Token number: 16384

Figure 15: Comparison of generated content between DEER and Vanilla on MATH-500.

2052

2053

2054

2055

2056

2057

2058

2059

2060

2061

2062

The expression $\$1 + 2 + 3 - 4 + 5 + 6\$$ is equivalent to 13. If we add parentheses in various locations, we can change this value. For example, $\$1+2+3-(4+5)+6=1+2+3-9+6=6-9+6=3\$$. If we only add parentheses (and not rearrange the order of the terms), what is the minimum possible value this expression can obtain? (For the purposes of this problem, we cannot change addition or subtraction into multiplication. For example, $\$(1+2+3)(-4)+5+6\$$ is not a valid placement of parentheses.)
Enter your answer as a single value.",

2063

2064

2065

2066

2067

2068

Answer:**DEER**

Okay, so I have this math problem here: the expression is $1 + 2 + 3 - 4 + 5 + 6$, and it's equal to 13. The task is to add parentheses in various places to change the value, but I can't rearrange the terms or turn the operations into multiplication. I need to find the minimum possible value this expression can obtain by adding parentheses. Hmm, interesting.
First, let me make sure I understand the problem correctly. I can only add parentheses, which will change the order in which operations are performed, but I can't move the numbers around or change the plus and minus signs into multiplication or division. So, for example, I can do something like $(1 + 2) + 3 - 4 + 5 + 6$, but I can't do $1*(2+3) - 4 + 5 + 6$ because that would involve multiplication, which isn't allowed.

<omitted>

Yep, that's correct. So, 13 is the original value.
Alternatively, group $(1 + 2) + (3 - 4 + 5 + 6)$. So, $1 + 2 = 3$, $3 - 4 = -1$, $-1 + 5 = 4$, $4 + 6 = 10$. Then, $3 + 10 = 13$.
Alternatively, group $1 + (2 + 3 - 4 + 5) + 6$. So, $2 + 3 = 5$, $5 - 4 = 1$, $1 + 5 = 6$. Then, $1 + 6 + 6 = 13$.
Hmm, seems like 3 is the lowest I can get so far. But let me think if there's another way.
What if I group $1 + 2 + 3 - (4 + 5 + 6)$. Let's compute that: $4 + 5 + 6 = 15$. Then, $1 + 2 + 3 = 6$, $6 - 15 = -9$. So, that's -9. Wait, that's way lower than 3. Is that allowed?

</thinks>

2084

2085

2086

2087

2088

2089

2090

2091

2092

2093

2094

2095

2096

2097

2098

2099

2100

2101

2102

2103

2104

2105

Token number: ~1500

Figure 16: Comparison of generated content between DEER and Vanilla on MATH-500.

Vanilla CoT

Okay, so I have this math problem here: the expression is $1 + 2 + 3 - 4 + 5 + 6$, and it's equal to 13. The task is to add parentheses in various places to change the value, but I can't rearrange the terms or turn the operations into multiplication. I need to find the minimum possible value this expression can obtain by adding parentheses. Hmm, interesting.
First, let me make sure I understand the problem correctly. I can only add parentheses, which will change the order in which operations are performed, but I can't move the numbers around or change the plus and minus signs into multiplication or division. So, for example, I can do something like $(1 + 2) + 3 - 4 + 5 + 6$, but I can't do $1*(2+3) - 4 + 5 + 6$ because that would involve multiplication, which isn't allowed.

<omitted>

Yep, that's correct. So, 13 is the original value.
Alternatively, group $(1 + 2) + (3 - 4 + 5 + 6)$. So, $1 + 2 = 3$, $3 - 4 = -1$, $-1 + 5 = 4$, $4 + 6 = 10$. Then, $3 + 10 = 13$.
Alternatively, group $1 + (2 + 3 - 4 + 5) + 6$. So, $2 + 3 = 5$, $5 - 4 = 1$, $1 + 5 = 6$. Then, $1 + 6 + 6 = 13$.
Hmm, seems like 3 is the lowest I can get so far. But let me think if there's another way.
What if I group $1 + 2 + 3 - (4 + 5 + 6)$. Let's compute that: $4 + 5 + 6 = 15$. Then, $1 + 2 + 3 = 6$, $6 - 15 = -9$. So, that's -9. Wait, that's way lower than 3. Is that allowed?

Wait, hold on. The original expression is $1 + 2 + 3 - 4 + 5 + 6$. If I group $4 + 5 + 6$, that would be subtracting 15, so $6 - 15 = -9$.

<omitted>

Wait, but in the original expression, after the subtraction, it's $+5 + 6$. So, if I group $4 + 5 + 6$, that would be changing the operations after the subtraction, but I can't rearrange terms. So, is that allowed?

<omitted>

Alternatively, group $1 + (2 + 3 - 4 + 5) + 6$. As before, $1 + 2 + 3 = 6$, $6 - 15 = -9$. So, that's -9. Wait, that's way lower than 3. Is that allowed?

Endless exhaustive search!**Token number: 16384**

2106
2107
2108
2109
2110
2111
2112
2113

2114 **Question:** The coordinates of a parallelogram are $(5, 3)$, $(6, 8)$, $(7, 4)$ and (x, y) and $x > 7$.
2115 What is the value of $x + y$?

2116 **Answer:**

Okay, so I have this problem where I need to find the coordinates of the fourth vertex of a parallelogram. The given points are $(5, 3)$, $(6, 8)$, $(7, 4)$, and (x, y) with the condition that $x > 7$. Then, I need to find the value of $x + y$. Hmm, let me think about how to approach this. First, I remember that in a parallelogram, opposite sides are equal and parallel. That means the midpoints of the diagonals should be the same. So, if I can find the midpoint of one diagonal, it should be equal to the midpoint of the other diagonal. Let me label the points to make it clearer. Let's say point A is $(5, 3)$, point B is $(6, 8)$, point C is $(7, 4)$, and point D is (x, y) . Now, I need to figure out which points are connected to each other. Since it's a parallelogram, the order of the points matters. I think the points should be connected in order, so A to B to C to D and back to A. But I'm not entirely sure, so maybe I should consider different possibilities. Alternatively, another way to approach this is by using vectors. In a parallelogram, the vector from one point to another should be equal to the vector from another point to the next. So, for example, vector AB should be equal to vector DC, and vector AD should be equal to vector BC. Let me try that. First, let's find vector AB. Point A is $(5, 3)$ and point B is $(6, 8)$. So, vector AB is $(6 - 5, 8 - 3) = (1, 5)$. If vector AB is equal to vector DC, then vector DC should also be $(1, 5)$. Point D is (x, y) and point C is $(7, 4)$. So, vector DC is $(7 - x, 4 - y)$. Setting this equal to $(1, 5)$, we get:
 $7 - x = 1 \Rightarrow x = 7 - 1 = 6$.
Solving the first equation: $7 - x = 1 \Rightarrow x = 7 - 1 = 6$.
Solving the second equation: $4 - y = 5 \Rightarrow y = 4 - 5 = -1$.

2125 <Induced trial answer>: $\boxed{6 + (-1)}$.  **Confidence: 0.7344** **Continue thinking**

2127 **Wait**, but the problem states that $x > 7$, and here x is 6, which is less than 7. That can't be right. So, maybe I assigned the points incorrectly.

2128 Perhaps the order of the points isn't A, B, C, D. Maybe it's a different order. Let me try another approach.
2129 Another property of a parallelogram is that the diagonals bisect each other. So, the midpoint of diagonal AC should be the same as the midpoint of diagonal BD.
2130 Let me calculate the midpoint of AC first. Point A is $(5, 3)$ and point C is $(7, 4)$. The midpoint formula is $((x_1 + x_2)/2, (y_1 + y_2)/2)$. So, midpoint of AC is $((5 + 7)/2, (3 + 4)/2) = (12/2, 7/2) = (6, 3.5)$.
2131 Now, the midpoint of BD should also be $(6, 3.5)$. Point B is $(6, 8)$ and point D is (x, y) . So, midpoint of BD is $((6 + x)/2, (8 + y)/2)$. Setting this equal to $(6, 3.5)$:
 $(6 + x)/2 = 6 \Rightarrow 6 + x = 12 \Rightarrow x = 6$.
Solving the first equation: $(6 + x)/2 = 6 \Rightarrow 6 + x = 12 \Rightarrow x = 6$.
Again, I get $x = 6$ and $y = -1$, but x is supposed to be greater than 7. Hmm, that's conflicting. Maybe I need to consider a different pair of points as the diagonal.

2134 <Induced trial answer>: $\boxed{10}$.  **Confidence: 0.5312** **Continue thinking**

2135 **Wait**, perhaps I assumed the wrong diagonal. Maybe the diagonals are AD and BC instead of AC and BD. Let me try that.

2136 Midpoint of AD: Point A is $(5, 3)$ and point D is (x, y) . Midpoint is $((5 + x)/2, (3 + y)/2)$.
2137 Midpoint of BC: Point B is $(6, 8)$ and point C is $(7, 4)$. Midpoint is $((6 + 7)/2, (8 + 4)/2) = (13/2, 12/2) = (6.5, 6)$.
2138 Setting midpoints equal:
 $(5 + x)/2 = 6.5 \Rightarrow 5 + x = 13 \Rightarrow x = 8$.
Solving the first equation: $(5 + x)/2 = 6.5 \Rightarrow 5 + x = 13 \Rightarrow x = 8$.
Okay, this gives $x = 8$ and $y = 9$, which satisfies $x > 7$. So, the fourth point is $(8, 9)$, and $x + y = 8 + 9 = 17$.

2139 <Induced trial answer>: $\boxed{17}$.  **Confidence: 0.9993** **Exit!**

2140 </think>

2142 To find the coordinates of the fourth vertex of the parallelogram, we use the property that the diagonals of a parallelogram bisect each other.

1. **Calculate the midpoint of diagonal BC:**
 $\text{Midpoint formula: } ((\frac{5+7}{2}, \frac{3+4}{2})) = ((\frac{12}{2}, \frac{7}{2})) = (6, 3.5)$.
2. **Set the midpoint of diagonal AD equal to the midpoint of BC:**
 $\text{Let D be } (x, y)$.
 $\text{Midpoint of AD: } ((\frac{5+x}{2}, \frac{3+y}{2})) = (6.5, 6)$.
 $\Rightarrow 5 + x = 13 \Rightarrow x = 8$.
 $\Rightarrow 3 + y = 12 \Rightarrow y = 9$.
3. **Calculate $x + y$:**
 $\Rightarrow x + y = 8 + 9 = 17$.
4. **Answer:** $\boxed{17}$

2148 Figure 17: An example demonstrating LRM's implicit ability to sense the emergence of pearl reasoning.
2149 When the model's reasoning content is insufficient to support the elicited answer, the confidence
2150 remains at a low level, as demonstrated by the first two confidence values in the figure. Conversely,
2151 when the model has provided adequate and sound reasoning as justification, the confidence reaches
2152 high levels, potentially approaching 1, as shown by the final confidence value in the figure.

2153
2154
2155
2156
2157
2158
2159

2160
2161
2162
2163
2164
2165
2166

2167 **Question:** Define $p = \sum_{k=1}^{\infty} \frac{1}{k^2}$ and $q = \sum_{k=1}^{\infty} \frac{1}{k^3}$. Find a way to write $\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{1}{(j+k)^3}$ in terms of p and q .

2170

DeepSeek-R1-Distill-Qwen-32B

Length

Okay, so I have this problem where I need to express the double sum $\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{1}{(j+k)^3}$ in terms of $\zeta(p)$ and $\zeta(q)$, where $\zeta(p) = \sum_{k=1}^{\infty} \frac{1}{k^p}$ and $\zeta(q) = \sum_{k=1}^{\infty} \frac{1}{k^q}$. Hmm, let me think about how to approach this. First, I know that $\zeta(3)$ is the famous Basel problem, which equals $\zeta(6)$, and $\zeta(5)$ is the Riemann zeta function at 5, which is known as Apery's constant, approximately 1.202... But I don't think I know the exact values; instead, I need to relate the double sum to these series. Looking at the double sum, it's over all positive integers $\zeta(j)$ and $\zeta(k)$, and the term is $\frac{1}{(j+k)^3}$. Maybe I can change variables to simplify this. Let me set $\zeta(n+j+k)$. Then, for each fixed $\zeta(n)$, $\zeta(j)$ can range from 1 to $\zeta(n-1)$, and $\zeta(k) = \zeta(n-j)$. So, perhaps I can rewrite the double sum as a single sum over $\zeta(n)$, where each term is the sum over $\zeta(j)$ from 1 to $\zeta(n-1)$ of $\frac{1}{(j+k)^3}$. Let me write that down: $\sum_{n=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{(j+k)^3} = \sum_{n=2}^{\infty} \sum_{j=1}^{n-1} \frac{1}{(j+k)^3}$. That makes sense because the smallest $\zeta(n)$ can be 2 (when $\zeta(j) = 1$) and $\zeta(k) = 1$), and it goes to infinity. Now, for each $\zeta(n)$, the inner sum is just adding $\frac{1}{(n-1)^3}$ a total of $\zeta(n-1)$ times. So, the inner sum simplifies to $\zeta(n-1)^3$. So now, the double sum becomes: $\sum_{n=2}^{\infty} \sum_{j=1}^{n-1} \zeta(n-1)^3$. Hmm, that seems manageable. Let me write this as: $\sum_{n=2}^{\infty} \sum_{j=1}^{n-1} \zeta(n-1)^3 = \sum_{n=2}^{\infty} \zeta(n-1)^3 \sum_{j=1}^{n-1} 1 = \sum_{n=2}^{\infty} \zeta(n-1)^3 (n-1)$. This is equal to: $\zeta(2)^3 \zeta(1) + \zeta(3)^3 \zeta(2) + \zeta(4)^3 \zeta(3) + \dots$. So, if I subtract the $\zeta(n-1)$ terms from both series, I can express the sum starting from $\zeta(2) = 2$. Let's compute that: For the first sum: $\sum_{n=2}^{\infty} \sum_{j=1}^{n-1} \zeta(n-1)^3 = \zeta(2)^3 \zeta(1) + \zeta(3)^3 \zeta(2) + \zeta(4)^3 \zeta(3) + \dots = 2 - \zeta(1)^3 = 2 - 1$. Similarly, for the second sum: $\sum_{n=2}^{\infty} \sum_{j=1}^{n-1} \zeta(n-1)^3 = \zeta(3)^3 - \zeta(4)^3 + \zeta(5)^3 - \zeta(6)^3 + \dots = p - 1 - q + 1 = p - q$.

So, the double sum simplifies to $(p - q)$. Therefore, the answer is $\boxed{p - q}$.

2183

15

Okay, so I need to find a way to express the double sum $\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{1}{(j+k)^3}$ in terms of $\langle p \rangle$ and $\langle q \rangle$, where $\langle p \rangle$ is the sum of $\langle 1/k^2 \rangle$ and $\langle q \rangle$ is the sum of $\langle 1/k^3 \rangle$. Hmm, let me think about how to approach this.
First, I remember that double sums can sometimes be simplified by changing variables or switching the order of summation. Maybe I can let $\langle n = j+k \rangle$, so that for each $\langle n \rangle$, the terms where $\langle j+k = n \rangle$ contribute to the sum. That might help me rewrite the double sum as a single sum over $\langle n \rangle$, with each term multiplied by the number of pairs $\langle (j, k) \rangle$ that add up to $\langle n \rangle$. Let me try that. Set $\langle n = j+k \rangle$. Then for each $\langle n \rangle \geq 2$ (since $\langle j \rangle$ and $\langle k \rangle$ are at least 1), the number of pairs $\langle (j, k) \rangle$ such that $\langle j+k = n \rangle$ is $\langle n-1 \rangle$. For example, when $\langle n = 2 \rangle$, there's only one pair $\langle 1,1 \rangle$; when $\langle n = 3 \rangle$, there are two pairs $\langle 1,2 \rangle$ and $\langle 2,1 \rangle$, and so on. So the double sum can be rewritten as: $\sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \frac{1}{n^3}$. Wait, is that right? Let me check. Each term $\langle \frac{1}{(j+k)^3} \rangle$ for each pair $\langle (j, k) \rangle$ would be grouped by $\langle n = j+k \rangle$, so each $\langle n \rangle$ contributes $\langle (n-1) \rangle$ terms each of $\langle \frac{1}{n^3} \rangle$. Therefore, the total contribution for each $\langle n \rangle$ is $\langle \frac{1}{n^3} \rangle$. So the entire double sum becomes: $\sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \frac{1}{n^3}$. Hmm, okay. Now I need to express this in terms of $\langle p \rangle$ and $\langle q \rangle$. Let me see. Let's first write out the sum: $\sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \frac{1}{n^3} = \sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \left[\frac{1}{n^2} - \frac{1}{(n-1)^2} \right] = \sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \left[\frac{1}{n^2} - \frac{1}{(n-1)^2} \right]$. Ah, that simplifies to the difference between two sums. Let's split that into two separate sums: $\sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \frac{1}{n^2} - \sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \frac{1}{(n-1)^2}$. Now, notice that these are just the tails of the series for $\langle p \rangle$ and $\langle q \rangle$. Since $\langle p \rangle = \sum_{k=1}^{\infty} \langle k \rangle \langle k^2 \rangle$, then $\langle \sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \frac{1}{n^2} \rangle = p - \langle \frac{1}{2} \rangle = p - 1$. Similarly, $\langle q \rangle = \sum_{k=1}^{\infty} \langle k \rangle \langle k^3 \rangle$, so $\langle \sum_{n=2}^{\infty} \langle n-2 \rangle \langle n \rangle \frac{1}{(n-1)^3} \rangle = q - \langle \frac{1}{1} \rangle = q - 1$. Therefore, substituting back into the expression: $\langle p - 1 - (q - 1) \rangle = p - q + 1 = \boxed{p - q + 1}$.

Let's summarize the above thought process and provide the result. The double sum can be rewritten by considering the sum over $\{n = j + k\}$. For each $\{n \geq 2\}$, there are $\{(n - 1)\}$ pairs $\{(j, k)\}$ such that $\{j + k = n\}$. This transforms the double sum into a single sum: $\sum_{j=1}^{\infty} \sum_{k=1}^{n-1} \frac{(-1)^{j+k}}{j+k}$. Breaking this into two separate sums: $\sum_{j=1}^{\infty} \sum_{k=2}^{n-1} \frac{(-1)^{j+k}}{j+k} - \frac{(-1)^{j+1}}{j+1}$. Recognizing these as the tails of the series for $\{(\text{p})\}$ and $\{(\text{q})\}$, we subtract the first terms: $\{(\text{p}) - 1 - (\text{q})\}$. Thus, the double sum simplifies to $\boxed{(\text{p} - \text{q})}$. Final Answer: $\boxed{(\text{p} - \text{q})}$

199
200
201
202

To express the double sum $\sum_{j=1}^n \sum_{k=1}^m \frac{1}{(j+k)^3}$ in terms of (p) and (q) , we start by changing variables. Let $(n = j + k)$. For each $(n \geq 2)$, the number of pairs (j, k) such that $(j + k = n)$ is $(n - 1)$. This allows us to rewrite the double sum as: $\sum_{n=2}^{\infty} \frac{1}{n^3} \sum_{j=1}^{n-1} \frac{1}{(j+1)^3}$. Next, we split this into two separate sums: $\sum_{n=2}^{\infty} \frac{1}{n^3} \left(\sum_{j=1}^{n-1} \frac{1}{(j+1)^3} - \sum_{j=1}^{n-1} \frac{1}{(j+2)^3} \right)$. Recognizing these as the tails of the series for (p) and (q) , we have: $\sum_{n=2}^{\infty} \frac{1}{n^3} \left(\frac{1}{n^3} - \frac{1}{(n+1)^3} \right) = p - \frac{1}{(p+1)^3}$. Substituting these into the expression gives: $p - \frac{1}{(p+1)^3} - (q - \frac{1}{(q+1)^3}) = p - q + \frac{1}{(p+1)^3} - \frac{1}{(q+1)^3}$. Thus, the double sum simplifies to $\boxed{p - q + \frac{1}{(p+1)^3} - \frac{1}{(q+1)^3}}$.

387

360

128

2204 Figure 18: An example demonstrating the presence of more than one `</think>` tokens in QwQ-32B.
 2205 For both models, the green thinking chunk is sufficient to draw the correct conclusion. However,
 2206 QwQ-32B proceeds with an additional summary (red chunk) and generates its own `</think>` token.
 2207 Based on all the above content, it arrives at the conclusion

2214
2215
2216
2217
2218
2219

Question: Convert the point $(0, 3)$ in rectangular coordinates to polar coordinates. Enter your answer in the form (r, θ) , where $r > 0$ and $0 \leq \theta < 2\pi$.

Original Response

Okay, so I need to convert the point $(0, 3)$ from rectangular coordinates to polar coordinates. Hmm, let me recall how polar coordinates work. In polar coordinates, a point is represented as (r, θ) , where r is the distance from the origin to the point, and θ is the angle measured from the positive x-axis to the line connecting the origin to the point. **<omitted>** So in this case, since the point is $(0, 3)$, θ is $\pi/2$. **But let me think again. <omitted>** **Wait, let me check if there's another way to think about this. <omitted>** **But let me just verify with another method. <omitted>** Just to make sure, let me think of another point.

For example, ... <omitted>

</think> To convert the rectangular coordinate point $(0, 3)$ to polar coordinates, we use the standard conversion formulas: **<omitted>** Step 1: Compute the radius r **<omitted>** Step 2: Determine the angle θ **<omitted>** Step 3: Final Answer **</think>** We are given the requirement that $r > 0$ and $0 \leq \theta < 2\pi$, which is satisfied here. **</think>** Thus, the polar coordinates of the point $(0, 3)$ are: **<omitted>** $\boxed{(3, \frac{\pi}{2})}$ **</think>**

DEER's Response

Okay, so I need to convert the point $(0, 3)$ from rectangular coordinates to polar coordinates. Hmm, let me recall how polar coordinates work. In polar coordinates, a point is represented as (r, θ) , where r is the distance from the origin to the point, and θ is the angle measured from the positive x-axis to the line connecting the origin to the point. **<omitted>** So in this case, since the point is $(0, 3)$, θ is $\pi/2$. **<Exit>**

</think> To convert the point $(0, 3)$ from rectangular coordinates to polar coordinates, we use the standard conversion formulas... **<omitted>** Final Answer: **<omitted>** $\boxed{(3, \frac{\pi}{2})}$ **</think>**

Less Tokens

Figure 19: Case1 for rebuttal

2230
2231
2232
2233
2234
2235
2236
2237

Question: Convert the point $(0, 3)$ in rectangular coordinates to polar coordinates. Enter your answer in the form (r, θ) , where $r > 0$ and $0 \leq \theta < 2\pi$.

DEER's Response

Okay, so I need to convert the point $(0, 3)$ from rectangular coordinates to polar coordinates. Hmm, let me recall how polar coordinates work. In polar coordinates, a point is represented as (r, θ) , where r is the distance from the origin to the point, and θ is the angle measured from the positive x-axis to the line connecting the origin to the point. **<omitted>** So in this case, since the point is $(0, 3)$, θ is $\pi/2$. **<Exit>**

</think> To convert the point $(0, 3)$ from rectangular coordinates to polar coordinates, we use the standard conversion formulas... **<omitted>** Final Answer: **<omitted>** $\boxed{(3, \frac{\pi}{2})}$ **</think>**

GPT5's Judgement

```
{
  "reasoning_sufficiency": "Sufficient",
  "justification": "The reasoning correctly identifies the goal: converting  $(0, 3)$  from rectangular to polar coordinates. The student recalls the definitions of  $r$  and  $\theta$  in polar form and explicitly writes the conversion formulas. They compute  $r$  step by step as  $\sqrt{0^2 + 3^2} = 3$ , which is consistent with their own formula. For  $\theta$ , they notice that the direct use of  $\arctan(y/x)$  fails when  $x = 0$  and explicitly acknowledge  $3/0$  is undefined. Instead of forcing the formula, they reason geometrically: the point  $(0, 3)$  lies on the positive y-axis, so the angle from the positive x-axis must be  $\pi/2$ . They also cross-check by contrasting with the case  $(0, -3)$ , which would give  $3\pi/2$ , showing awareness of quadrant/axis dependence. All necessary components—distance from origin, correct handling of the undefined tangent case, and correct angle selection within the required range—are present and logically connected to the final answer  $(3, \pi/2)$ ."
}
```

Figure 20: LLM-as-a-judge result for rebuttal

2265
2266
2267

