Benchmarking Deep Thinking and Reasoning of Large Language Models

Anonymous ACL submission

Abstract

Recent advancements in large language models (LLMs) have led to the development of large reasoning models (LRMs), which incorporate intermediate deep thinking to guide decision-making. These LRMs have demonstrated promising results in a range of domains, including commonsense reasoning, mathematics, and code generation. However, the precise role of deep thinking in improving model performance remains underexplored, and no universally accepted framework exists to evaluate its impact. To address this gap, we introduce DeepThinkBench, a comprehensive benchmarking framework designed to evaluate the effects of deep thinking on instruction-based LLMs. Our experiments reveal three key findings: 1) incorporating deep thinking from LRMs significantly enhances the performance of instruction-based LLMs, particularly in tasks that require multi-step reasoning; 2) deep thinking improves both accuracy and efficiency, though the extent of improvement varies depending on the task; and 3) we propose three distinct rankings (i.e., ranking single LLMs, ranking single LRMs, and ranking combined LLMs) providing a holistic view of deep thinking. These contributions highlight the potential of integrating deep thinking to advance instruction-based LLM capabilities, and we advocate for further research on optimizing deep thinking integration to enhance model scalability, robustness, and real-world applicability across diverse tasks.

1 Introduction

Recent advancements in large language models (LLMs), such as OpenAI-o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025), and Gemini-2.0-flash (Abacha et al., 2024), have significantly advanced the field of large reasoning models (LRMs). These models leverage a process known as "Deep Thinking," wherein the model explicitly generates intermediate reasoning steps to guide its inferences. This approach has demonstrated significant improvements in tasks requiring multi-step problem-solving, including commonsense reasoning (Davis, 2023), machine translation (Wang et al., 2022), and natural language understanding (Dong et al., 2019). Notably, LRMs excel in tasks involving logical reasoning, offering more sophisticated and structured responses compared to traditional LLMs.

Despite these advancements, LRMs still encounter challenges, particularly in balancing computational efficiency with performance across diverse and complex tasks. Although these LLMs exhibit strong reasoning abilities, issues related to response time, scalability, and resource consumption remain significant (King, 2022). Additionally, while integrating deep thinking into LRMs has been explored, its potential integration into general-purpose instruction-based LLMs remains underexplored (Jiang et al., 2025). Therefore, the central focus of this study is to investigate how deep thinking can enhance the performance of instruction-based LLMs. Specifically, we seek to answer the following research question: **How does the integration of deep thinking, represented by the intermediate thoughts of LRMs, impact the accuracy and efficiency of instruction-based LLMs across various tasks?**

To address this question, we introduce **DeepThinkBench**, a benchmarking framework designed to systematically evaluate how deep thinking enhances the performance of instruction-based LLMs. The framework consists of three key stages: 1) extracting deep thinking from LRMs across various reasoning tasks, capturing the multi-step reasoning processes; 2) constructing structured prompts for instruction-based LLMs, integrating the deep thinking into their reasoning processes; and 3) evaluating the performance of the instruction-based LLMs across multiple tasks to assess the effect of integrating deep thinking.

To comprehensively evaluate this effect, we con-

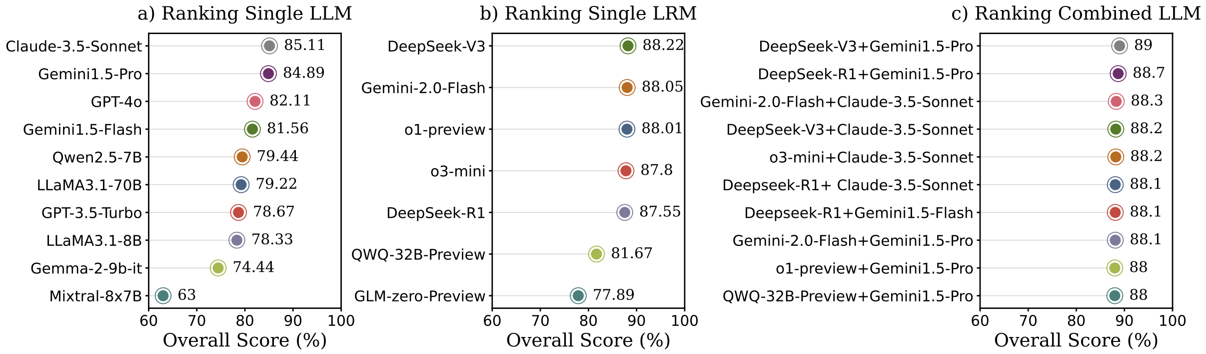

Figure 1: Rankings based on the DeepThinkBench evaluation. These rankings provide insights into the performance of different models and highlight the best-performing LLMs and LRMs in each category.

struct three specialized datasets: *DeepThinkBench-base*, a general-purpose dataset with 1,000 question-answer pairs from 10 diverse datasets, covering 5 reasoning tasks (details in Table 1); *DeepThinkBench-think*, which contains deep thinking examples; and *DeepThinkBench-fewshot*, focusing on few-shot examples. Our experiments demonstrate that incorporating deep thinking from LRMs significantly improves the performance of instruction-based LLMs, particularly for tasks requiring multi-step reasoning. However, this integration also leads to decreased efficiency, with more intricate tasks showing greater variability in performance improvement compared to simpler tasks.

Ranking. To present our findings in a comprehensive and intuitive manner, we introduce three ranking categories, as illustrated in Figure 1: a) Ranking Single LLM: A comprehensive evaluation of instruction-based LLMs based on their overall performance. b) Ranking Single LRM: A comprehensive assessment of LRMs based on their overall performance. c) Ranking Combined LLM: An evaluation of instruction-based LLMs after integrating deep thinking, measuring the impact of reasoning integration. These rankings highlight the top-performing models in each category and provide a foundation for further research on optimizing the integration of LRMs to improve model performance in real-world tasks (detailed results are provided in Appendix B).

Contributions. In summary, this paper makes the following contributions:

- **A Comprehensive Benchmark:** We introduce DeepThinkBench, a novel framework for assessing the impact of deep thinking on instruction-based LLMs. The framework facilitates detailed comparisons between instruction-based and LRMs across a wide range of tasks.
- **Three Datasets:** We develop three specialized datasets (i.e., *DeepThinkBench-base*, *DeepThinkBench-think*, and *DeepThinkBench-fewshot*) to evaluate reasoning processes of LLMs, performance improvements with deep thinking, and efficiency in few-shot learning scenarios. These datasets provide valuable insights into how deep thinking influences instruction-based LLMs.
- **Insights and Implications:** Our work offers new perspectives on the interaction between instruction-based LLMs and LRMs, showing how integrating deep thinking can improve instruction-following tasks. The findings, alongside our ranking categories, provide significant implications for future research on model integration and the practical deployment of LLMs in reasoning-intensive tasks. Notably, the rankings we introduce in this paper (Figure 3 and Table B) serve as a critical contribution, offering a method to evaluate the effectiveness of deep thinking integration and optimize future LLM development.

2 DeepThinkBench

DeepThinkBench is a comprehensive benchmarking framework developed to assess the impact of deep thinking, facilitated by large reasoning models (LRMs), on the performance of instruction-based LLMs across a variety of tasks. As shown in Figure 2, the framework involves three key stages: (1) extracting deep thinking from LRMs, (2) constructing prompts for instruction-based LLMs, and (3) evaluating performance across multiple tasks.

2.1 Step 1: Thought Extraction from LRMs

The first step of the framework focuses on extracting reasoning processes, referred to as deep think-

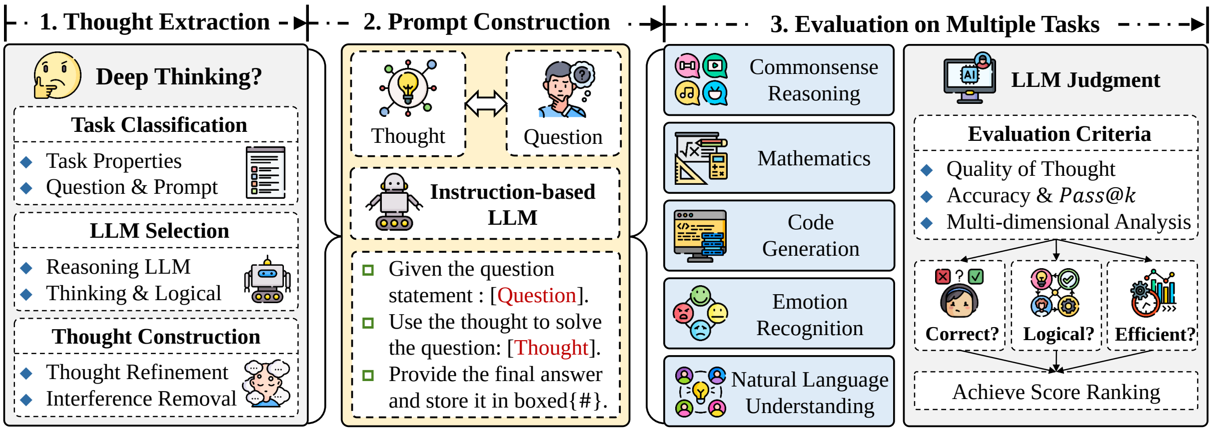

Figure 2: Overview of the DeepThinkBench framework. 1) Thought Extraction, focusing on the generation of deep thinking; 2) Prompt Construction, mainly for integrating deep thinking; 3) Evaluation on multiple tasks.

ing, from LRMs. The process begins with task classification, where the properties of the task, such as CommonSense (Davis and Marcus, 2015) and Math (Barroso et al., 2021), are analyzed to derive the most appropriate questions and prompts for guiding the reasoning process. Next, one LRM (e.g., OpenAI-o1-Preview (Zhong et al., 2024), DeepSeek-R1 (Guo et al., 2025)) is selected to generate intermediate reasoning steps that follow logically from the task at hand. These intermediate steps are then refined, with irrelevant or extraneous information removed, ensuring that only the necessary reasoning steps are retained.

2.2 Step 2: Prompt Construction for Instruction-based LLMs

The second step involves using the extracted thoughts to construct structured prompts for instruction-based LLMs (e.g., LLaMA3.1-8B-Instruct (Dubey et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024b)). The formulated question, based on the classification and task analysis, is presented to the LLM. The deep thinking (i.e., the intermediate thoughts generated by the LRM) is then incorporated into the prompt to help the LLM follow a coherent reasoning path throughout the task. These prompts provide the necessary context, enabling the LLM to engage in multi-step reasoning that mirrors the thought process previously extracted from the reasoning LRM. Once the prompt is processed by the LLM, a response is generated and subsequently evaluated for logical consistency and adherence to the reasoning steps.

2.3 Step 3: Evaluation on Multiple Tasks

The third step evaluates the response generated by the instruction-based LLM across diverse reasoning tasks, including commonsense reasoning (Sap et al., 2020), mathematics (Hendrycks et al., 2021), code generation (Li et al., 2022), emotion recognition (Coronado et al., 2019), and natural language understanding (NLU) (Dua et al., 2019). Multiple performance metrics are used to assess the output, including the quality of thoughts (deep thinking), accuracy, and pass@k. A multi-dimensional analysis of these results is conducted, considering logical consistency, efficiency, and overall correctness. The final rankings provide a holistic view of the impact of deep thinking on instruction-based LLM performance across different tasks.

2.4 Research Questions

This study investigates the following key research questions:

[RQ1] How does the integration of deep thinking compare between single LRMs and combined LLMs in terms of overall performance?
[RQ2] What is the impact of incorporating deep thinking into LLMs on their accuracy across a variety of datasets?
[RQ3] What is the relationship between the length of deep thinking and the accuracy of LLMs across different datasets?
[RQ4] How does the length of deep thinking influence the response time of LLMs, and is there a positive correlation?
[RQ5] How do zero-shot and few-shot learning conditions affect the performance improvements facilitated by deep thinking in LLMs?

Table 1: Summary of datasets used in the experiments, detailing dataset size, domain, evaluation methods, and metrics. It highlights both standard and custom datasets for evaluating various reasoning capabilities.

Dataset	Dataset Information			Evaluation Method	
	Size	Domain	Deep Thinking	Multi-Dimensional	Metric
OpenBookQA (Mihaylov et al., 2018)	5,957	Commonsense Reasoning	✗	✗	Accuracy
HellaSwag (Zellers et al., 2019)	39,905	Commonsense Reasoning	✗	✗	Accuracy
GSM8K (Chen et al., 2024)	8,792	Mathematics	✗	✗	Accuracy
MATH (Hendrycks et al., 2021)	12,500	Mathematics	✗	✗	Accuracy
MBPP (Austin et al., 2021)	974	Code Generation	✗	✗	Pass@k
HumanEval (Chen et al., 2021)	164	Code Generation	✗	✗	Pass@k
SST-2 (Coronado et al., 2019)	67,349	Emotion Recognition	✗	✗	Accuracy
IMDB (Dodds, 2006)	100,000	Emotion Recognition	✗	✗	Accuracy
SQUAD (Rajpurkar et al., 2018)	98,169	NLU	✗	✗	Accuracy
DROP (Dua et al., 2019)	86,945	NLU	✗	✗	Accuracy
DeepThinkBench-base	1,000	General	✗	✓	Overall Score
DeepThinkBench-think	1,000	Deep Thinking	✓	✓	Overall Score
DeepThinkBench-fewshot	100	Few-shot Example	✓	✗	Quality Score

3 Experiment Settings

Large Reasoning Models. LRMs are equipped with the ability to generate deep thinking during task execution. The models considered in this study include QwQ-32B[1] (Yang et al., 2024a), o1-preview (Zhong et al., 2024), o3-mini (OpenAI, 2025), DeepSeek-R1 (Guo et al., 2025), DeepSeek-V3 (Liu et al., 2024), Gemini-2.0-Flash (Google, 2025), and GLM-Zero-Preview (ZhipuAI, 2025). These models demonstrate strong analytical capabilities and consistency across a range of reasoning tasks. Further details about these models are provided in Appendix A.

Instruction-based LLMs. To systematically investigate the impact of deep thinking on instruction-based LLMs, we incorporate a range of LLMs, spanning different scales and sources. These models include Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024), LLaMA3.1-8B-Instruct (Dubey et al., 2024), LLaMA3.3-70B-Instruct (Dubey et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024a), Gemma-2-9b-It (Team et al., 2024), Gemini-1.5-Flash (Reid et al., 2024), Gemini-1.5-Pro (Reid et al., 2024), Claude-3.5-Sonnet (Ahtropic, 2024), GPT-3.5-Turbo (Ye et al., 2023), and GPT-4o (Hurst et al., 2024). Further details about LLMs are also provided in Appendix A.

Datasets. We utilize a diverse set of datasets to evaluate the reasoning capabilities of the models across different domains. A summary of the datasets used in this study is provided in Table 1. We constructed three distinct datasets: (1) *DeepThinkBench-base*, a general-purpose dataset integrating a variety of reasoning tasks (i.e., commonsense reasoning, mathematics, code generation, emotion recognition, and natural language understanding), including OpenBookQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019), GSM8K (Chen et al., 2024), MATH (Hendrycks et al., 2021), MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021), SST-2 (Coronado et al., 2019), IMDB (Dodds, 2006), SQUAD (Rajpurkar et al., 2018), and DROP (Dua et al., 2019); (2) *DeepThinkBench-think*, which contains deep-thinking examples generated by all LRMs for each question in the *DeepThinkBench-base*; (3) *DeepThinkBench-fewshot*, containing 10 few-shot examples generated by the o3-mini-high model (OpenAI, 2025) for each dataset in *DeepThinkBench-base*.

Evaluation Metrics. To quantify the effectiveness of deep thinking integration and assess model performance, we utilize several key evaluation metrics, including:

- **Accuracy (ACC):** This metric measures the proportion of correct predictions made by the model. It is computed by dividing the number of correct predictions by the total number of predictions, providing a straightforward assessment of model performance.
- **Pass@k:** This metric evaluates the percentage of times the correct answer appears among the top-k predictions generated by the model. This

[1]This type of 'Preview' and 'Instruct' may be omitted from the diagram, and so on for all LLMs

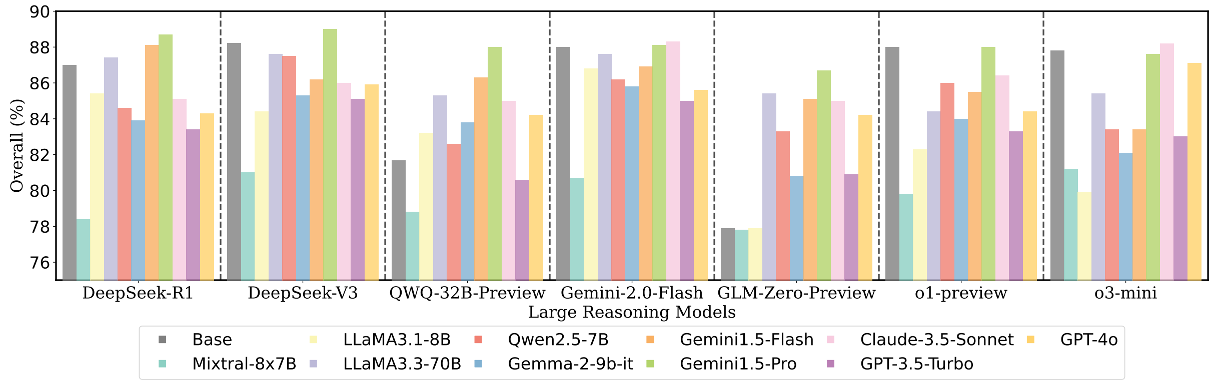

Figure 3: Comparison of performance between single LRM and combined LLMs. Each bar represents the performance of a combined LLM utilizing deep thinking from its respective LRM. The overall ranking details are available in Appendix B.

is particularly useful in tasks (i.e., code generation), and the correct answer is expected to appear within the top-k results.

- **Quality Score:** Assessed through *LLM-as-a-judge*, this metric evaluates the relevance, logical consistency, completeness, fluency, and depth of thought in the *deep thinking*.
- **Overall Score:** This composite score aggregates various task-specific metrics, including ACC and Pass@k, to provide an overall performance measure.

4 Empirical Results and Analysis

This section provides a comprehensive analysis of the experimental results, focusing on how deep thinking derived from LRMs influences the performance of LLMs across various tasks.

4.1 Effect of Deep Thinking on Single LRM and Combined LLMs

[RQ1] How does the integration of deep thinking compare between single LRMs and combined LLMs in terms of overall performance?
Conclusion: Combined LLMs outperform single LRMs overall. A ranking for all combined LLMs has been created to identify the optimal LRM-LLM combinations in B.

To evaluate the impact of deep thinking, we compare the performance of LLMs in two configurations: the single LRM mode, where LRMs operate independently on *DeepThinkBench-base*, and the combined LLM mode, where deep thinking from LRMs is applied to LLMs on *DeepThinkBench-think*. As depicted in Figure 3, the results indicate the following: 1) the combined LLMs, which integrate deep thinking, generally outperform the single LRM models. This suggests that instruction-based LLMs, with their stronger instruction-following abilities, benefit more from incorporating deep thinking. 2) The varying performance across different combined LLMs underscores that optimal deep thinking strategies may vary depending on the LLM used. 3) The figure facilitates a comparison of the relative performance of different combined LLMs, offering insights into which combinations of LRMs and instruction-based LLMs yield the best results. The specific rankings can be found in Appendix B.

4.2 Effect of Deep Thinking on Accuracy for diverse Datasets

[RQ2] How does incorporating deep thinking into LLMs affect their accuracy across various datasets?
Conclusion: Deep thinking significantly improves the accuracy of LLMs, particularly for those with lower initial capabilities and for complex tasks requiring multi-step reasoning.

To examine how deep thinking impacts accuracy across different datasets, we compare the performance of LLMs before and after integrating deep thinking. Table 15 presents the results across 8 datasets, showing that deep thinking consistently improves the accuracy of LLMs. Notably, *Mixtral-8x7B* experienced the most significant performance gain, whereas models like *Gemini1.5-Pro* and *Claude-3.5-Sonnet* showed minimal improvements. This trend suggests that weaker LLMs benefit more from deep thinking, possibly due to their lower initial performance. Moreover, datasets that involve more complex reasoning tasks, such

Table 2: Impact of Deep Thinking on Accuracy for LLMs. The table shows the improvement in average accuracy across 8 datasets for LLMs after incorporating deep thinking. ‘Base’ refers to *DeepThinkBench-base*, and ‘DT’ represents *DeepThinkBench-think*. Further details on each LLM’s performance can be found in Appendix C.

LLM	Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Mixtral-8x7B	Base	83.00	31.00	77.00	27.00	25.00	75.00	88.00	72.00
	DT	88.14 (+5.14)	61.17 (+30.17)	88.43 (+11.43)	70.50 (+43.50)	37.67 (+12.67)	90.50 (+15.50)	88.83 (+16.83)	87.20 (+15.20)
LLaMA3.1-8B	Base	86.00	56.00	95.00	47.00	66.00	86.00	92.00	86.00
	DT	88.14 (+2.14)	56.17 (+0.17)	90.29 (+4.29)	59.83 (+12.83)	75.50 (+9.50)	89.50 (+3.50)	93.33 (+1.33)	92.40 (+6.40)
LLaMA3.1-70B	Base	92.00	62.00	98.00	68.00	35.00	94.00	91.00	82.00
	DT	89.86 (-2.14)	65.83 (+3.83)	91.86 (-6.14)	83.83 (+15.83)	75.83 (+40.83)	91.33 (-2.67)	92.83 (+1.83)	87.60 (+5.60)
Qwen2.5-7B	Base	86.00	53.00	95.00	44.00	79.00	92.00	92.00	82.00
	DT	88.29 (+2.29)	60.50 (+7.50)	94.43 (-0.57)	66.17 (+22.17)	82.50 (+3.50)	90.50 (-1.50)	91.83 (-0.17)	89.80 (+7.80)
Gemma-2-9b-it	Base	84.00	52.00	90.00	44.00	62.00	82.00	84.00	84.00
	DT	88.00 (+4.00)	60.50 (+8.50)	93.43 (+3.43)	73.67 (+29.67)	72.50 (+10.50)	90.00 (+8.00)	91.00 (+7.00)	87.20 (+3.20)
Gemini1.5-Flash	Base	86.00	42.00	93.00	77.00	79.00	85.00	95.00	87.00
	DT	86.86 (+0.86)	58.17 (+16.17)	94.43 (+1.43)	81.83 (+4.83)	83.33 (+4.33)	89.83 (+4.83)	92.17 (+2.17)	90.20 (+3.20)
Gemini1.5-Pro	Base	91.00	57.00	96.00	80.00	86.00	88.00	90.00	87.00
	DT	88.71 (-2.29)	64.67 (+7.67)	96.57 (+0.57)	85.00 (+5.00)	90.00 (+4.00)	90.67 (+2.67)	93.67 (+3.67)	89.80 (+2.80)
Claude-3.5-Sonnet	Base	92.00	53.00	99.00	72.00	84.00	93.00	93.00	86.00
	DT	88.00 (-4.00)	67.50 (+14.50)	92.43 (-6.57)	78.67 (+6.67)	75.50 (-8.50)	90.33 (-2.67)	95.17 (+2.17)	93.60 (+7.60)
GPT-3.5-Turbo	Base	86.00	60.00	88.00	45.00	72.00	81.00	91.00	70.00
	DT	90.00 (+4.00)	58.00 (-2.00)	91.71 (+3.71)	73.50 (+28.50)	58.67 (-12.33)	91.17 (+10.17)	91.33 (+0.33)	89.80 (+19.80)
GPT-4o	Base	92.00	66.00	94.00	49.00	81.00	87.00	93.00	86.00
	DT	86.71 (-5.29)	64.00 (-2.00)	95.00 (+1.00)	65.17 (+16.17)	77.00 (-4.00)	91.33 (+4.33)	94.83 (+1.83)	93.80 (+7.80)

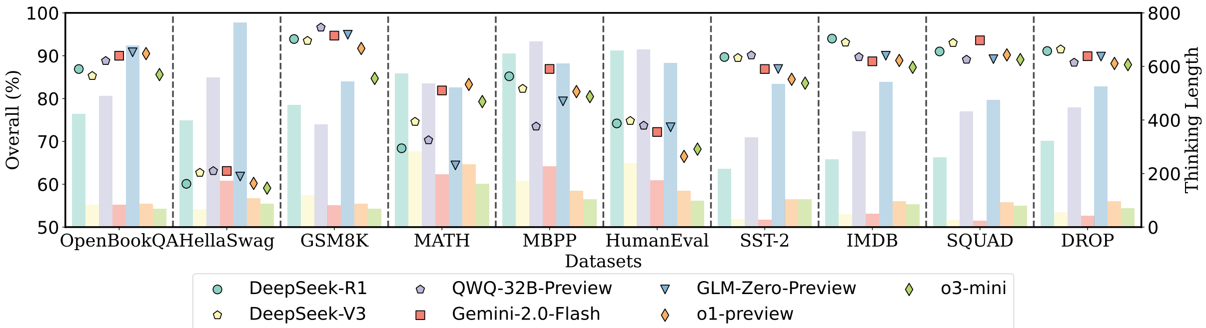

Figure 4: Impact of Deep Thinking Length on Accuracy. The bar chart shows the deep thinking length for different LRMs across datasets, while the scatter plot illustrates the average accuracy of each LLM after applying deep thinking of various lengths. Detailed results for each LLM are available in Appendix F.

as *HellaSwag* and *MATH*, displayed the most noticeable accuracy improvements, highlighting that deep thinking is especially effective for complex tasks requiring multi-step reasoning. For specific performance details across datasets, please refer to Appendix F.

4.3 Impact of Deep Thinking Length on Accuracy

As illustrated in Figure 4, the results suggest that while longer reasoning steps generally improve performance on complex tasks, there is no consistent positive relationship between deep thinking length and overall accuracy. For example, *Gemini-2.0-Flash* showed better accuracy with shorter deep thinking on *HellaSwag*, and *o1-preview* outperformed others with shorter reasoning on *OpenBookQA*. On the other hand, for more complex datasets like *MATH*, variations in deep thinking length had more significant impacts on performance, suggesting that the effectiveness of deep thinking length depends on the complexity of the task rather than a simple direct relationship.

[RQ3] What is the relationship between the length of deep thinking and the accuracy of LLMs across different datasets?
Conclusion: Deep thinking length influences accuracy, but the relationship is not always positive, where task complexity plays a significant role.

4.4 Impact of Deep Thinking Length on Response Time

We further examine how the length of deep thinking affects the response time of LLMs. As depicted in Figure 5, there is a positive correlation between deep thinking length and response time,

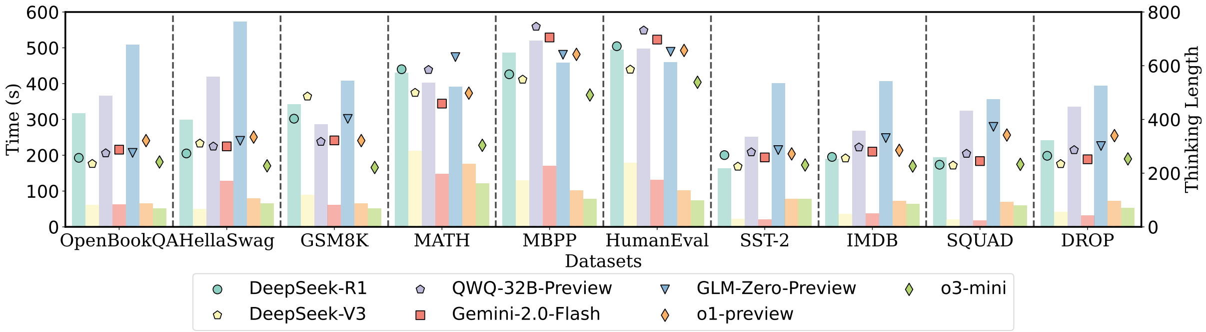

Figure 5: Impact of Deep Thinking Length on Response Time. The bar chart shows the deep thinking length for different LRMs, and the scatter plot represents the response time for each LLM after applying deep thinking. Detailed information is provided in Appendix C.

meaning that longer reasoning steps generally result in slower responses. This trend is consistent across datasets, including *OpenBookQA*, *SST-2*, and *DROP*, with noticeable delays in response time as deep thinking length increases. Even in more complex tasks like *MATH*, the relationship between reasoning length and response time remains evident, suggesting that longer reasoning chains require more computational resources and lead to increased processing time.

[RQ4] How does the length of deep thinking affect the response time of LLMs? Is there a positive correlation?
Conclusion: Longer deep thinking steps are associated with slower response times, particularly for tasks that require complex reasoning, such as code generation and mathematics.

4.5 Impact of Deep Thinking for Zero-shot and Few-shot Learning

We investigate the effect of deep thinking by comparing zero-shot and few-shot learning conditions. As shown in Table 3, the results indicate that deep thinking does not consistently improve performance under few-shot conditions. While *MATH* showed a slight improvement, most other datasets exhibited a decrease in accuracy when few-shot learning was applied. This suggests that when instruction-based LLMs receive both deep thinking and additional few-shot examples, the cognitive load becomes too high, negatively affecting performance. Notably, **Mixtral-8x7B** performed the worst under few-shot conditions, likely due to its limited instruction-following capabilities.

Table 3: Zero-shot vs. Few-shot accuracy for different LLMs. DT represents deep thinking (zero-shot), and Few represents deep thinking (few-shot).

LLM	Type	HellaSwag	MATH	IMDB
Mixtral-8x7B	DT	61.16	70.50	92.50
	Few	(51.40 (-9.76))	(64.60 (-5.90))	(83.80 (-8.70))
LLaMA3.1-8B	DT	56.16	59.83	91.50
	Few	(61.60 (+5.44))	(80.80 (+20.97))	(90.00 (-1.50))
LLaMA3.1-70B	DT	65.83	83.83	93.66
	Few	(64.80 (-1.03))	(85.40 (+1.57))	(92.40 (-1.26))
Qwen2.5-7B	DT	60.50	66.16	93.33
	Few	(62.80 (+2.30))	(81.40 (+15.24))	(92.00 (-1.33))
Gemma-2-9b-it	DT	60.50	73.66	90.33
	Few	(58.80 (-1.70))	(74.00 (+0.34))	(91.80 (+1.47))
Gemini1.5-Flash	DT	58.16	81.83	93.16
	Few	(59.20 (+1.04))	(81.80 (-0.03))	(92.60 (-0.56))
Gemini1.5-Pro	DT	64.66	85.00	90.83
	Few	(63.40 (-1.24))	(85.60 (+0.60))	(91.80 (+0.97))
Claude-3.5-Sonnet	DT	67.50	78.66	92.83
	Few	(62.60 (-4.90))	(79.60 (+0.94))	(92.60 (-0.23))
GPT-3.5-Turbo	DT	58.00	73.50	92.83
	Few	(61.00 (+3.00))	(74.00 (+0.50))	(90.20 (-2.63))
GPT-4o	DT	64.00	65.16	90.33
	Few	(63.20 (-0.80))	(83.40 (+18.24))	(90.20 (-0.13))

[RQ5] How do zero-shot and few-shot affect the performance improvements facilitated by deep thinking in LLMs?
Conclusion: Few-shot learning conditions do not enhance performance significantly, except in complex tasks where deep thinking can help leverage additional examples effectively.

4.6 Judge for the Quality of Deep Thinking

To evaluate the quality of deep thinking, we use DeepSeek-R1 as a judge, applying five criteria: relevance, logical consistency, completeness, fluency, and depth of thought. The results, as shown in Table 4, reveal that Gemini-2.0-Flash produced the highest average quality deep thinking across all criteria. Other LRMs exhibited varying performance, with some excelling in certain areas (e.g., fluency and completeness) but lagging behind in others. Combined with Figure 1, we find that the quality of deep thinking is positively correlated with the overall performance of LLMs on reasoning tasks.

Table 4: LLM-as-a-Judge of Deep Thinking Quality for LRMs. Each column represents a criterion. **Rel** refers to relevance, **Log.** to logical, **Cpt.** to completeness, **Flc.** to fluency, and **Depth.** to Depth of Thought.

Reasoning LLM	Rel.	Log.	Cpt.	Flc.	Depth.	Avg.
QWQ-32B-Preview	7	8	8	8	7	7.6
o1-preview	8	**9**	8	8	8	8.2
o3-mini	7	8	8	8	8	7.8
DeepSeek-R1	8	8	**9**	8	**9**	8.4
DeepSeek-V3	8	**9**	8	8	7	8
Gemini-2.0-Flash	**9**	8	**9**	**9**	**9**	**8.8**
GLM-Zero-Preview	8	8	7	8	7	7.6

5 Related Work

Exploring the Reasoning Abilities of LLMs. Recent studies on Large Language Models (LLMs) have increasingly focused on improving their reasoning capabilities, particularly for complex, multi-step tasks. Various strategies have been proposed to enhance both the accuracy and efficiency of reasoning in LLMs (Savage et al., 2024; Liu et al., 2025; Yang et al., 2024c). Reward models, such as Outcome-Based Reward Models (ORMs) (Uesato et al., 2022; Yu et al., 2023) and Process-Based Reward Models (PRMs) (Zeng et al., 2021), aim to guide reasoning processes by offering feedback on areas for improvement throughout the reasoning chain (Havrilla et al., 2024; Setlur et al., 2024). Furthermore, research on multi-agent discussions has highlighted the advantages of collaborative reasoning, though it has also been found that strong single-agent prompts can often produce comparable results (Wang et al., 2024). Another notable line of research emphasizes the role of reasoning memory, with studies suggesting that categorizing premises into determinate and indeterminate types allows for more refined and accurate reasoning over time (Sun et al., 2024). In addition, efforts to enhance the robustness and interpretability of reasoning processes have led to the development of automated reasoning evaluation tools and standardized libraries (Hao et al., 2024). Until the emergence of o1-preview (Temsah et al., 2024) and DeepSeek-R1 (Guo et al., 2025), marking the beginning of the era for large reasoning models (LRMs).

In-Depth Exploration of LRMs. A defining characteristic of Large Reasoning Models (LRMs) is their ability to generate intermediate reasoning steps, a process referred to as deep thinking. This involves models explicitly articulating their intermediate thought processes to guide decision-making. Recent LRMs, such as the QwQ-32B-Preview (Yang et al., 2024a), o1 and o3 series (Jaech et al., 2024; OpenAI, 2025), DeepSeek series (Guo et al., 2025; Liu et al., 2024) , Gemini-2.0 series (Google, 2025) (referenced in 3), as well as Kimi-v1.5 long-CoT, and Kimi-v1.5 short-CoT, incorporate deep thinking strategies. These models either generate extended chains of reasoning or more concise reasoning steps, depending on the specific task requirements. However, there is a notable gap in the exploration of the effects of deep thinking in these models. This paper aims to establish a benchmark for evaluating these effects.

6 Future Directions

Future research will focus on two key directions: (1) exploring a broader set of combinations between LRMs and LLMs and (2) investigating the role of LRMs as judges for evaluating the quality of deep thinking. While this study has provided valuable insights into the impact of deep thinking on LLM performance, expanding the range of LRM-LLM pairs explored across diverse tasks will help identify optimal combinations for specific applications. Additionally, examining the use of LRMs as evaluators of deep thinking quality will offer a more comprehensive understanding of how different LRMs contribute to enhancing the reasoning capabilities of LLMs.

7 Conclusion

This paper introduces DEEPTHINKBENCH, a novel benchmarking framework for evaluating the impact of deep thinking on LLM performance. Our analysis demonstrates that integrating deep thinking from reasoning LLMs significantly improves LLM accuracy, especially for complex tasks requiring multi-step reasoning. Beyond the evaluation framework, we introduce three ranking categories (i.e., Single LLM, Single LRM, and Combined LLM), offering a comprehensive view of model performance across different configurations. These rankings provide insights into the effectiveness of deep thinking integration and guide future efforts to optimize LLM performance in reasoning-intensive tasks, underscoring the potential of deep thinking to advance LLM capabilities.

Limitations

This study has several limitations. First, we evaluated a limited set of LRMs and LLMs, so the results may not generalize to other models or tasks. Further research with a broader range of models is needed to validate the scalability of our findings. Additionally, while deep reasoning improves accuracy, it may lead to slower response times for tasks requiring extensive reasoning. Future work should focus on optimizing reasoning efficiency to balance accuracy and real-time performance.

References

Asma Ben Abacha, Wen-wai Yim, Yujuan Fu, Zhaoyi Sun, Meliha Yetisgen, Fei Xia, and Thomas Lin. 2024. Medec: A benchmark for medical error detection and correction in clinical notes. *arXiv preprint arXiv:2412.19260*.

Claude Ahtropic. 2024. Claude. [Online]. Available: https://www.anthropic.com/claude.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Connie Barroso, Colleen M Ganley, Amanda L McGraw, Elyssa A Geer, Sara A Hart, and Mia C Daucourt. 2021. A meta-analysis of the relation between math anxiety and math achievement. *Psychological bulletin*, 147(2):134.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*.

Michael J Coronado, Katelyn A Bruno, Lori A Blauwet, Carsten Tschöpe, Madeleine W Cunningham, Sabine Pankuweit, Sophie Van Linthout, Eun-Seok Jeon, Dennis M McNamara, Jan Krejčí, et al. 2019. Elevated sera sst 2 is associated with heart failure in men 50 years old with myocarditis. *Journal of the American Heart Association*, 8(2):e008968.

Ernest Davis. 2023. Benchmarks for automated commonsense reasoning: A survey. *ACM Computing Surveys*, 56(4):1–41.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

Klaus Dodds. 2006. Popular geopolitics and audience dispositions: James bond and the internet movie database (imdb). *Transactions of the Institute of British Geographers*, 31(2):116–130.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Google. 2025. Gemini-2.0-flash.

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

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, et al. 2024. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *arXiv preprint arXiv:2404.05221*.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*.

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

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

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanwei Li, Yu Qi, Xinyan Chen, Liuhui Wang, Jianhan Jin, Claire Guo, Shen Yan, Bo Zhang, Chaoyou Fu, Peng Gao, and Hongsheng Li. 2025. Mme-cot: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency.

Carey W King. 2022. Interdependence of growth, structure, size and resource consumption during an economic growth cycle. *Biophysical Economics and Sustainability*, 7(1):1.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Jun Xia, Tianyi Wu, Zhiwei Xue, Yulin Chen, Kenji Kawaguchi, Jiaheng Zhang, and Bryan Hooi. 2025. Guardreasoner: Towards reasoning-based llm safeguards. *arXiv preprint arXiv:2501.18492*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

OpenAI. 2025. Openai-o3.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv*, abs/2403.05530.

Maarten Sap, Vered Shwartz, Antoine Bosselut, Yejin Choi, and Dan Roth. 2020. Commonsense reasoning for natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 27–33.

Thomas Savage, Ashwin Nayak, Robert Gallo, Ekanath Rangan, and Jonathan H Chen. 2024. Diagnostic reasoning prompts reveal the potential for large language model interpretability in medicine. *NPJ Digital Medicine*, 7(1):20.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2024. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9828–9862.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Mohamad-Hani Temsah, Amr Jamal, Khalid Alhasan, Abdulkarim A Temsah, and Khalid H Malki. 2024. Openai o1-preview vs. chatgpt in healthcare: a new frontier in medical ai reasoning. *Cureus*, 16(10).

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Haifeng Wang, Hua Wu, Zhongjun He, Liang Huang, and Kenneth Ward Church. 2022. Progress in machine translation. *Engineering*, 18:143–153.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *arXiv preprint arXiv:2402.18272*.

An Yang, Baosong Yang, Binyuan Hui, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024b. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Menglin Yang, Aosong Feng, Bo Xiong, Jiahong Liu, Irwin King, and Rex Ying. 2024c. Enhancing llm complex reasoning capability through hyperbolic geometry. In *ICML 2024 Workshop on LLMs and Cognition*.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *ArXiv*, abs/2303.10420.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Zhiguo Zeng, Yi-Ping Fang, Qingqing Zhai, and Shijia Du. 2021. A markov reward process-based framework for resilience analysis of multistate energy systems under the threat of extreme events. *Reliability Engineering & System Safety*, 209:107443.

ZhipuAI. 2025. Glm-zero-preview.

Tianyang Zhong, Zheng Liu, Yi Pan, et al. 2024. Evaluation of openai o1: Opportunities and challenges of agi. *ArXiv*, abs/2409.18486.

A Details of selected LLMs

We have investigated Large Language Models (LLMs) based on their performance in reasoning and instruction tracking tasks. These models fall into two categories: inference-based LLMs are designed for complex logic and reasoning tasks, while instruction-based LLMs are optimized for natural language instructions in tasks such as question answering and summarization. These models vary in size and ease of use, with some providing open weights to improve transparency, while others remain closed. Together, they represent the leading advances in natural language processing for reasoning and instruction tracking applications.

Table 5: The Details of selected LLMs

Category	**Model**	**Model Size**	**Version**	**Open-Weight?**	**Creator**
	QWQ-32B-Preview	32B	N/A	✓	Qwen
	OpenAI-o1-Preview	N/A	N/A	✗	OpenAI
	OpenAI-o3-mini	N/A	N/A	✗	OpenAI
Reasoning LLM	DeepSeek-R1	671B	N/A	✓	DeepSeek
	DeepSeek-V3	671B	N/A	✓	DeepSeek
	Gemini-2.0-Falsh	N/A	N/A	✗	Google
	GLM-Zero	N/A	N/A	✗	Zhipu AI
	Mixtral-8x7B	8x7B	instruct-v0.1	✓	Mistral
	LLaMA-3.1-8B	8B	instruct	✓	Meta
	LLaMA-3.1-70B	70B	instruct	✓	Meta
	Qwen2.5-7B	7B	N/A	✓	Qwen
Instruction-based LLM	Gemma-2-9b-It	9B	N/A	✓	Google
	Gemini-1.5-Flash	N/A	N/A	✗	Google
	Gemini-1.5-Pro	N/A	N/A	✗	Google
	Claude-3.5-Sonnet	N/A	20240620	✗	Anthropic
	GPT-3.5-Turbo	N/A	0125	✗	OpenAI
	GPT-4o	N/A	2024-08-06	✗	OpenAI

B Details of Results for Ranking of LRMs

This section records the complete Ranking information for all LRMs combined with LLMs, as shown in the table below, in descending order of overall score.

LRM	LLM	Score
Deepseek-V3	Gemini1.5-Pro	89
Deepseek-R1	Gemini1.5-Pro	88.7
Gemini-2.0-Flash	Claude-3.5-Sonnet	88.3
Deepseek-V3	Claude-3.5-Sonnet	88.2
o3-mini	Claude-3.5-Sonnet	88.2
Deepseek-R1	Claude-3.5-Sonnet	88.1
Deepseek-R1	Gemini1.5-Flash	88.1
Gemini-2.0-Flash	Gemini1.5-Pro	88.1
o1-preview	Gemini1.5-Pro	88
QwQ-32B	Gemini1.5-Pro	88
Deepseek-V3	LLaMA3.3-70B	87.6
Gemini-2.0-Flash	LLaMA3.3-70B	87.6
o3-mini	Gemini1.5-Pro	87.6
Deepseek-V3	Qwen2.5-7B	87.5
Deepseek-R1	LLaMA3.3-70B	87.4
o3-mini	GPT-4o	87.1
Gemini-2.0-Flash	Gemini1.5-Flash	86.9
Gemini-2.0-Flash	LLaMA3.1-8B	86.8
GLM-zero-preview	Gemini1.5-Pro	86.7
o1-preview	Claude-3.5-Sonnet	86.4
QwQ-32B	Gemini1.5-Flash	86.3
Deepseek-V3	Gemini1.5-Flash	86.2
Gemini-2.0-Flash	Qwen2.5-7B	86.2
o1-preview	Qwen2.5-7B	86
Deepseek-V3	GPT-4o	85.9
Gemini-2.0-Flash	Gemma-2-9b-it	85.8
Gemini-2.0-Flash	GPT-4o	85.6
o1-preview	Gemini1.5-Flash	85.5
Deepseek-R1	LLaMA3.1-8B	85.4
GLM-zero-preview	LLaMA3.3-70B	85.4
o3-mini	LLaMA3.3-70B	85.4
Deepseek-V3	Gemma-2-9b-it	85.3
QwQ-32B	LLaMA3.3-70B	85.3
Deepseek-V3	GPT-3.5	85.1
GLM-zero-preview	Gemini1.5-Flash	85.1
QwQ-32B	Claude-3.5-Sonnet	85
Gemini-2.0-Flash	GPT-3.5	85
GLM-zero-preview	Claude-3.5-Sonnet	85
Deepseek-R1	Qwen2.5-7B	84.6
Deepseek-V3	LLaMA3.1-8B	84.4
o1-preview	LLaMA3.1-70B	84.4
o1-preview	GPT-4o	84.4
Deepseek-R1	GPT-4o	84.3
QwQ-32B	GPT-4o	84.2
GLM-zero-preview	GPT-4o	84.2
o1-preview	Gemma-2-9b-it	84
Deepseek-R1	Gemma-2-9b-it	83.9
QwQ-32B	Gemma-2-9b-it	83.8
Deepseek-R1	GPT-3.5	83.4
o3-mini	Qwen2.5-7B	83.4
o3-mini	Gemini1.5-Flash	83.4
GLM-zero-preview	Qwen2.5-7B	83.3
o1-preview	GPT-3.5	83.3
QwQ-32B	LLaMA3.1-8B	83.2
o3-mini	GPT-3.5	83
QwQ-32B	Qwen2.5-7B	82.6
o1-preview	LLaMA3.1-8B	82.3
o3-mini	Gemma-2-9b-it	82.1
o3-mini	Mixtral-8x7B	81.2
Deepseek-V3	Mixtral-8x7B	81
GLM-zero-preview	GPT-3.5	80.9
GLM-zero-preview	Gemma-2-9b-it	80.8
Gemini-2.0-Flash	Mixtral-8x7B	80.7
QwQ-32B	GPT-3.5	80.6
o3-mini	LLaMA3.1-8B	79.9
o1-preview	Mixtral-8x7B	79.8
QwQ-32B	Mixtral-8x7B	78.8
Deepseek-R1	Mixtral-8x7B	78.4
GLM-zero-preview	LLaMA3.1-8B	77.9
GLM-zero-preview	Mixtral-8x7B	77.8

C Details of Deep Thinking Length on Accuracy

This section presents the findings of our experiment, where we evaluated the performance of 10 distinct Large Language Models (LLMs) across 10 datasets. Each of these LLMs, which were instruction-based due to being given deep thinking processes, utilized reasoning processes generated by 7 different reasoning models. These processes were analyzed to explore the impact of Thinking Length—defined as the number of reasoning steps or depth of the reasoning process—on the accuracy of model predictions. Contrary to our initial expectations, the results revealed that there was no straightforward relationship between Thinking Length and accuracy. In fact, increasing the Thinking Length did not consistently result in improved accuracy. For example, when evaluating the Mixtral-8x7B model on the MATH dataset, shorter Thinking Lengths produced higher accuracy scores compared to longer Thinking Lengths. This finding suggests that more extensive reasoning steps do not necessarily enhance the model's ability to produce accurate results. In other cases, even significant variations in Thinking Length had minimal impact on accuracy, indicating that the length of reasoning played a negligible role in determining performance. Moreover, in scenarios where Thinking Length was varied substantially, accuracy remained relatively stable, further reinforcing the notion that deeper reasoning does not always lead to better performance. These findings underscore the complex relationship between reasoning complexity (as represented by Thinking Length) and accuracy, indicating that other factors may have a more significant influence on model performance. Further research is needed to investigate these underlying factors and their implications for the design and optimization of LLMs.

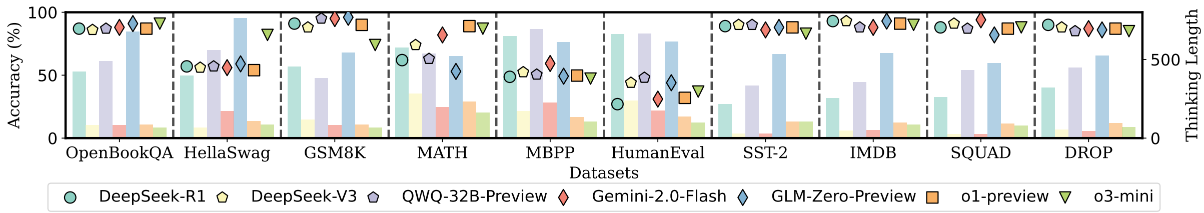

Figure 6: Effect of Deep Thinking Length on Mixtral-8x7B Accuracy

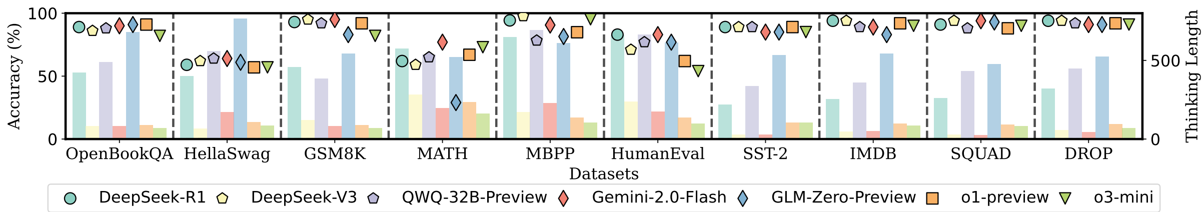

Figure 7: Effect of Deep Thinking Length on LLaMA3.1-8B Accuracy

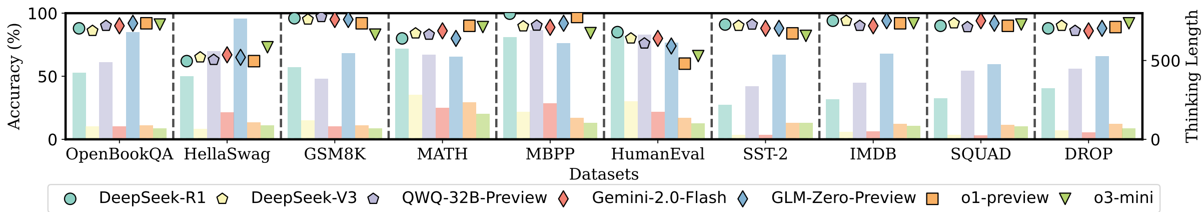

Figure 8: Effect of Deep Thinking Length on LLaMA3.1-70B Accuracy

D Details of Deep Thinking Length on Response Time

This section explores the impact of Deep Thinking Length on the response time of Instruction-Based LLMs, based on the same 10 models used in the previous experiment. However, in this case, we tested 6 distinct reasoning models across 10 datasets. The results showed that, similar to the previous findings on accuracy, there was no clear correlation between Deep Thinking Length and response time. For instance,

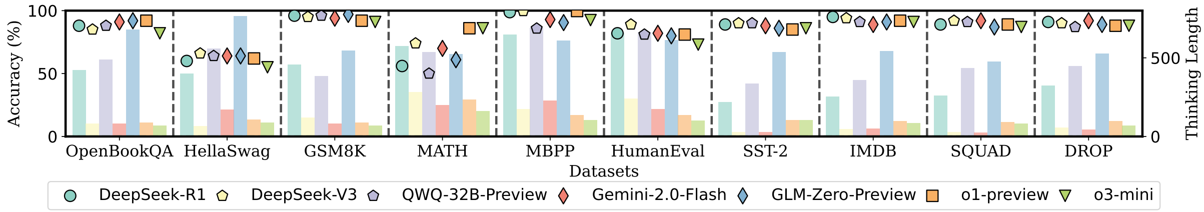

Figure 9: Effect of Deep Thinking Length on Qwen2.5-7B Accuracy

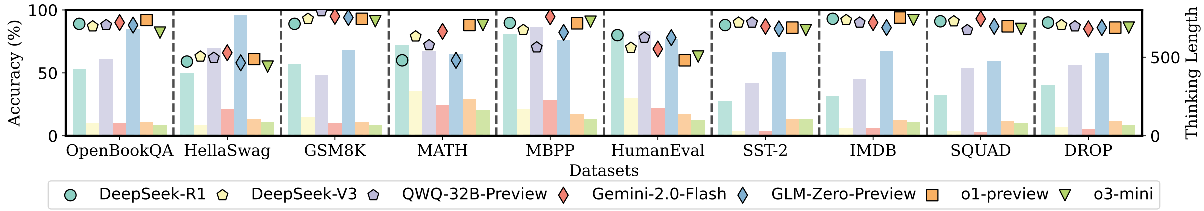

Figure 10: Effect of Deep Thinking Length on Gemma-2-9b-it Accuracy

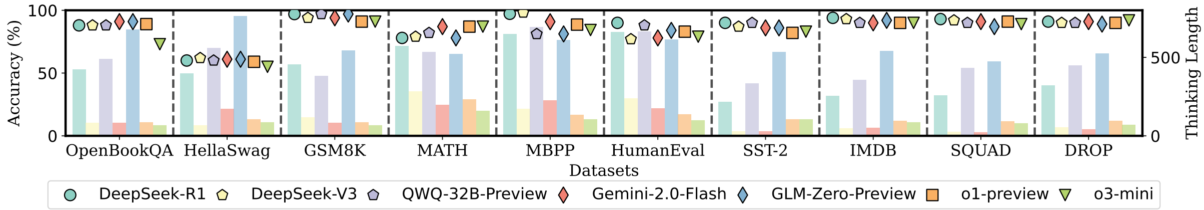

Figure 11: Effect of Deep Thinking Length on Gemini1.5-Flash Accuracy

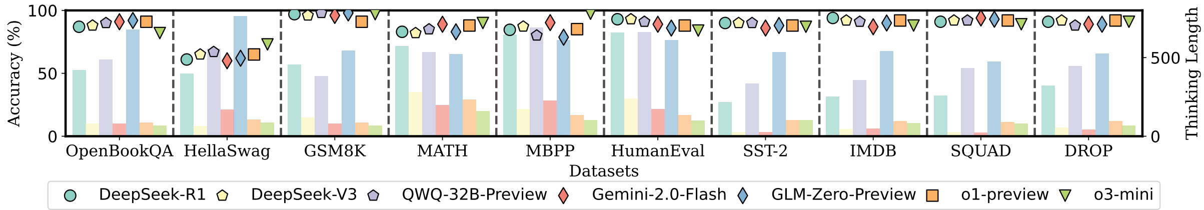

Figure 12: Effect of Deep Thinking Length on Gemini1.5-Pro Accuracy

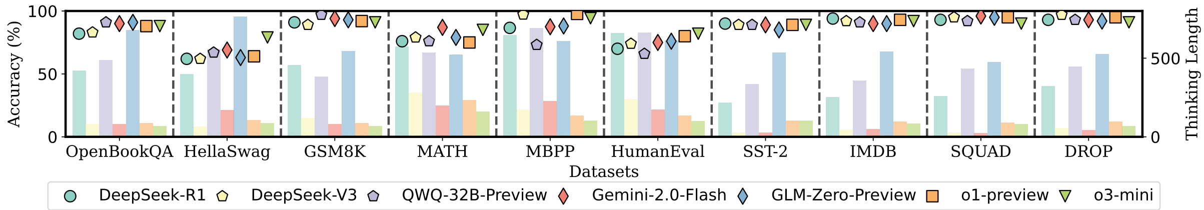

Figure 13: Effect of Deep Thinking Length on Claude-3.5-Sonnet Accuracy

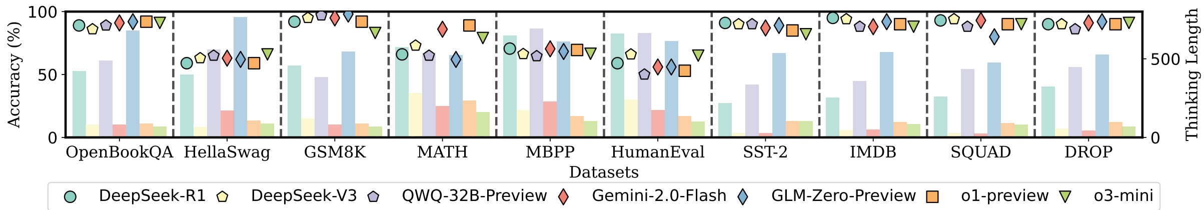

Figure 14: Effect of Deep Thinking Length on GPT-3.5-Turbo Accuracy

when evaluating the Qwen2.5-7B model on the MATH dataset, we observed that despite the Deep Thinking being relatively short for the o1-preview, the response time was significantly longer. In contrast, the Gemini-2.0-Flash model, which also used a relatively short Thinking Length, produced noticeably shorter response times. These findings suggest that the relationship between the depth of reasoning and the time taken for the model to generate a response is more complex than initially anticipated. Across other datasets, the response time varied unpredictably, with instances where shorter reasoning steps led to longer response times and vice versa. This reinforces the notion that Deep Thinking Length does not have a consistent or direct impact on the time it takes for an Instruction-Based LLM to generate a response. Further investigation is needed to identify the factors influencing response time, which may involve model-specific optimizations or other variables not captured by the length of reasoning alone.

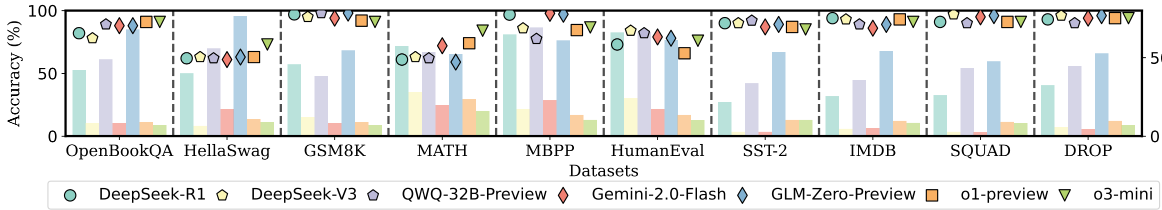

Figure 15: Effect of Deep Thinking Length on GPT-4o Accuracy

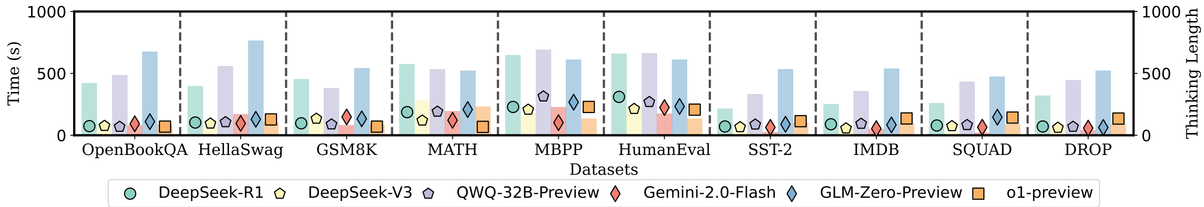

Figure 16: Effect of Deep Thinking Length on Mixtral-8x7B Response Time

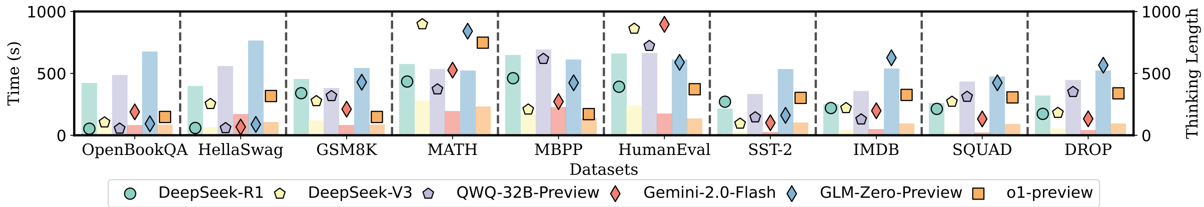

Figure 17: Effect of Deep Thinking Length on LLaMA3.1-8B Response Time

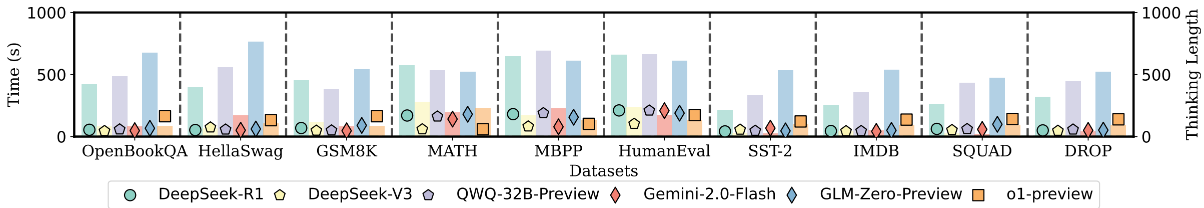

Figure 18: Effect of Deep Thinking Length on LLaMA3.1-70B Response Time

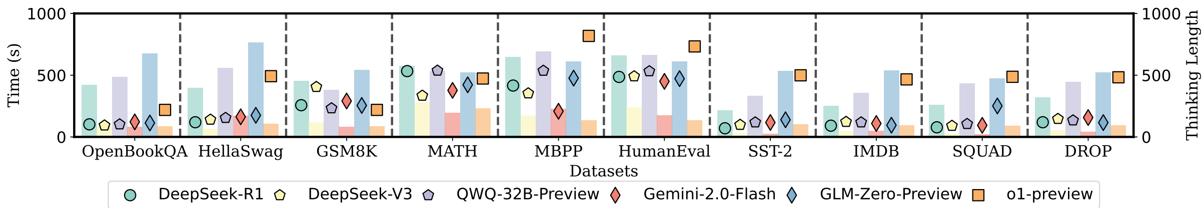

Figure 19: Effect of Deep Thinking Length on Qwen2.5-7B Response Time

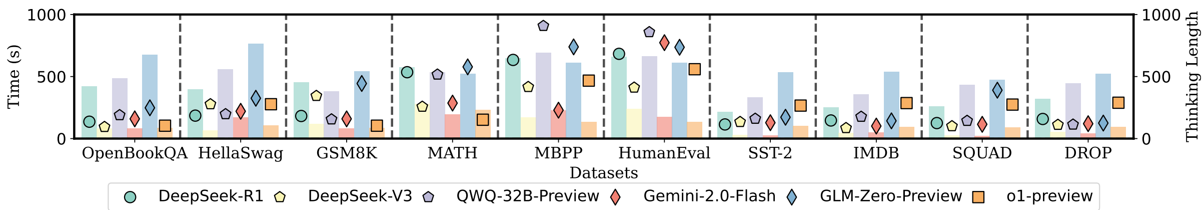

Figure 20: Effect of Deep Thinking Length on Gemma-2-9b-it Response Time

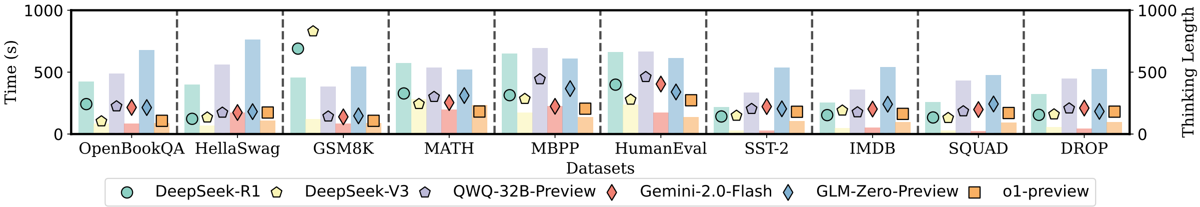

Figure 21: Effect of Deep Thinking Length on Gemini1.5-Flash Response Time

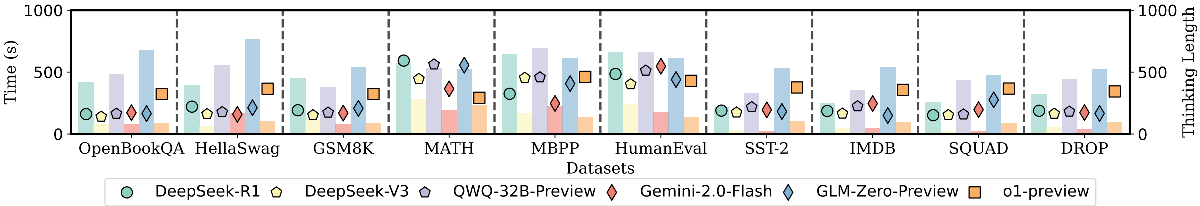

Figure 22: Effect of Deep Thinking Length on Gemini1.5-Pro Response Time

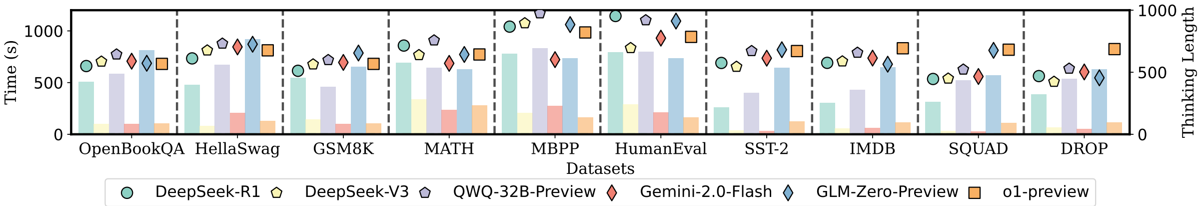

Figure 23: Effect of Deep Thinking Length on Claude-3.5-Sonnet Response Time

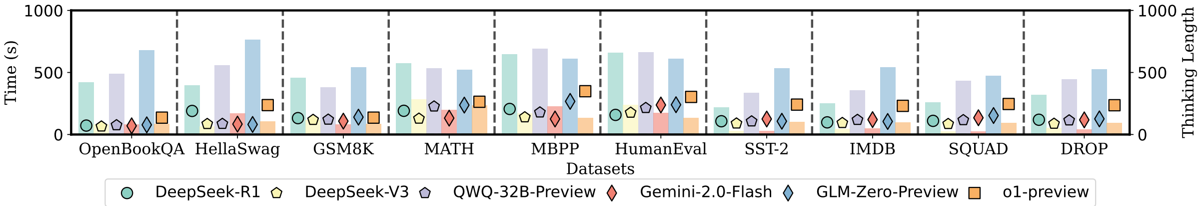

Figure 24: Effect of Deep Thinking Length on GPT-3.5-Turbo Response Time

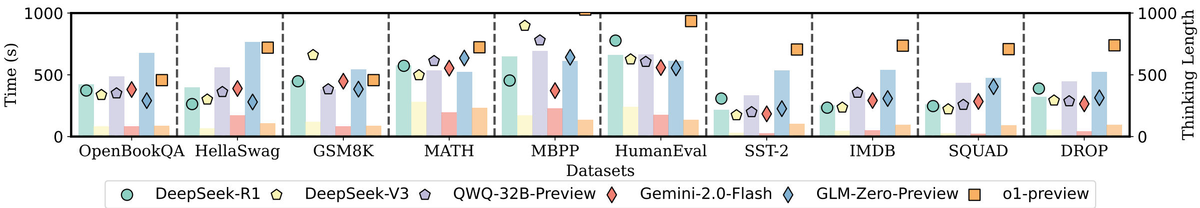

Figure 25: Effect of Deep Thinking Length on GPT-4o Response Time

E Details of Deep Thinking Examples

Deep Thinking. OpenBookQA's example is as follows:

OpenBookQA dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking
So I've got this question about Louis Pasteur. It says, L̈ouis Pasteur found a way to...änd then it gives four options: make food tasty, make food healthier, make food wet, or make food green. I need to pick the most appropriate one.First, I need to remember who Louis Pasteur was and what he's famous for. I know that Pasteur was a French biologist and chemist who lived in the 19th century. He made some really important contributions to science, especially in the areas of microbiology and vaccination.One of his most significant discoveries is pasteurization, which is a process of heating liquids to a specific temperature for a certain period to kill harmful bacteria without altering the liquid's quality, taste, or nutritional value. This process is widely used in the food industry, particularly for milk and wine.So, considering that pasteurization is about making food safer by eliminating harmful bacteria, I think option B, m̈ake food healthier,ïs the most appropriate choice. While pasteurization does help preserve the taste of food, that's not its primary purpose. It's mainly about making food safer to consume by reducing the risk of foodborne illnesses.Option A, m̈ake food tasty,m̈ight be a secondary benefit since pasteurization helps maintain the food's original taste by preventing spoilage, but it's not about enhancing taste. Option C, m̈ake food wet,d̈oesn't make much sense in this context. Option D, m̈ake food green,ïs also irrelevant to Pasteur's work.Therefore, the best answer is B: make food healthier.**Final Answer**[boxed{B}]

OpenAI-o1-preview's Deep Thinking
Harming an organism can lead to a decrease in its population. When organisms are harmed, they may suffer injuries or death, reducing the number of individuals in a population. This can negatively impact the population size over time.Therefore, the most appropriate option is: boxed{D}.

Deepseek-R1's Deep Thinking
<think>Okay, let's see. The question is about why seasons occur. The options are A. lazy, B. jaded, C. slanted, D. stilted.Hmm, I remember that seasons are caused by the Earth's axial tilt. So the axis is tilted, which makes different parts of the Earth get more or less sunlight at different times of the year. The question mentions that something huge slammed into the planet, which I think refers to the giant impact hypothesis where a Mars-sized body collided with Earth, leading to the formation of the Moon and causing the axial tilt. So the answer should be the word that means tilted. Looking at the options: A. Lazy 2̆013 that means not active or energetic. Doesn't fit here. B. Jaded 2̆013 means tired or bored. Not related to tilt. C. Slanted 2̆013 that means at an angle, tilted. That seems right. D. Stilted 2̆013 means stiff or unnatural, like in speech. Not related to axis tilt. So the correct answer should be C. Slanted. Let me double-check. Yes, the Earth's axis is slanted or tilted, which causes the seasons. The other options don't make sense in this context. So I'm confident it's C.</think>The occurrence of seasons is due to Earth's axial tilt. The question references a historical impact that caused this tilt. Among the options, s̈lanted(̈C) directly describes the axis being angled, aligning with the scientific explanation. Other options (lazy, jaded, stilted) are unrelated to tilt. boxed{C}.

Deepseek-V3's Deep Thinking
The correct answer is **D. zapping power**. When one electrical conductor contacts another, electrical current (often referred to as z̈apping power)̈ flows through both conductors.boxed{D}.

Gemini-2.0-preview's Deep Thinking
Birds require water to live. Dirt and soil can hold water, but they are not living things themselves. Rocks do not require water to live. Therefore, the answer is birds. boxed{A}.

GLM-zero-preview's Deep Thinking

###Thinking Let's tackle this question about bloodhounds. We know that bloodhounds are renowned for their exceptional sense of smell, which they use primarily for tracking. Their noses are so powerful that they can follow a scent trail that's days old and even detect odors that are underground or underwater. So, other than sight, how do bloodhounds find a meal?Let's consider the options:- **Social media**: Wait, bloodhounds don't have the cognitive ability to use social media, nor do they have the physical capability to operate electronic devices. This option seems highly unlikely.- **Their phone**: Similarly, bloodhounds can't use a phone. They don't understand the concept of phones or have the dexterity to operate them. This option doesn't make sense either.- **The internet**: Bloodhounds don't have access to the internet, and even if they did, they wouldn't know how to use it to find food. This option is also out of the question.- **Stench**: Now, this one makes sense! Bloodhounds are famous for their acute sense of smell. They can detect even the faintest odors, which would include the scent of food. Their powerful noses allow them to track down a meal by following the stench.Let's double-check the other options to be sure. Social media, phone, and internet are all technologies that require human understanding and manipulation. Bloodhounds don't have the cognitive ability to use these tools, and they certainly don't need them to find food. They've been tracking scents successfully long before humans invented these technologies.While we might joke about a bloodhound using social media to s̈mellp̈osts or find food, that's not how their sense of smell works. They don't have the capacity to connect scents to online information. Similarly, the idea of a bloodhound using a phone or the internet is purely fanciful.So, the most logical and appropriate answer is s̈tench.B̈loodhounds can find a meal by using their incredible sense of smell to detect the odor of food.**Answer:** [boxed{D}]###Response1. **Understanding the Question**-**Objective:** Determine how bloodhounds find a meal aside from using sight.- **Options:**- A. Social media - B. Their phone- C. The internet- D. Stench2. **Analyzing the Options**- **Social Media (Option A):**- Bloodhounds lack the cognitive ability to understand or use social media. - They cannot operate electronic devices. - **Conclusion:** Unlikely. - **Phone (Option B):**- Bloodhounds cannot use a phone due to lack of understanding and physical dexterity. - **Conclusion:** Unlikely. - **Internet (Option C):** Bloodhounds have no access to the internet. - They cannot navigate or utilize online resources to find food.- **Conclusion:** Unlikely.- **Stench (Option D):** - Bloodhounds are renowned for their exceptional sense of smell. - They can detect faint odors and track scent trails effectively. - **Conclusion:** Most plausible.3. **Final Determination** - Among all options, s̈tenchïs the most appropriate method by which bloodhounds can find a meal.**Answer:** [boxed{D}].

Deep Thinking. HellaSwag's example is as follows:

HellaSwag dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking

So I have this text here that seems to be about exploring and understanding desires. It's a bit fragmented, but I'll try to make sense of it. The main idea is to find a way to observe and explore desires, and it suggests a few methods like mindfulness, meditation, or just relaxing and watching the mind. The goal is to observe desires as they unfold and analyze their effects.First, it emphasizes setting clear guidelines for the practice, making sure not to get too involved with the desires or dreams that arise. So, it's about maintaining a certain distance and objectivity while observing.Option 1 suggests that when you observe the mind, you'll realize latent desires, and it mentions giving ẗhat name some weight.Ï think ẗhat namer̈efers to the desire when it arises. So, perhaps it's about acknowledging and labeling the desire when it comes up, giving it importance or attention in a mindful way.Option 2 talks about seeing yourself doing something without thinking about the outcome, meaning that the action is performed without being attached to a specific result. This could be related to performing a mindfulness exercise where the focus is on the process rather than the outcome.Option 3 seems a bit unclear; it mentions lying by reading and describing an object sliding off a ceiling, which might be a metaphor or a specific mindfulness exercise, but it's not very straightforward.Given these options, I think the most reasonable scenario is Option 1: When you observe things as they unfold, your mind will realize the latent desire. So, by practicing mindfulness or meditation, you become more aware of underlying desires that you might not have been conscious of otherwise. By observing these desires without judgment or attachment, you can better understand them and their impact on your thoughts and actions.Therefore, the final answer is:[boxed{text{Option 1: When you observe things as they unfold, your mind will realize the latent desire.}}].

Deepseek-R1's Deep Thinking
<think>Okay, let's try to figure out the right answer here. The user is asking about how to get discounts on Amazon, and there are several steps and options provided. Let me go through each part carefully.First, the header is Ḧow to get discounts on amazon,änd the title mentions choosing products by discount level. The step says to go to tjshome.com/greatdeals.php and click on the percentage like 50% or 90% in the shopping category. The substep then mentions navigating to a new window where all products are 50% or 90% off. The example talks about low-value merchandise being sold off and picking items that are öff the charts,ẅhich I think means heavily discounted.Now, there are three options given for the outcome. Let's look at each one:1. Add a special discount on items you like. Those items will get significantly lesser than what were advertised on Amazon.2. Click on the d̈iscountsẗab in the left column. If not open, click on the s̈toreẗab in the right column.3. Add the item to your cart to automatically receive the discount. The title also mentions going to cheapsally.com for current Amazon discounts.Hmm. The original steps involve going to an external site (tjshome.com) to find deals by percentage. The first option talks about adding a special discount, making items cheaper than advertised. That might not make sense because Amazon's advertised prices are usually fixed unless there's a coupon or promo code. But the step here is about using an external site to find already discounted items, not applying additional discounts.The second option mentions clicking on a discounts tab. But the original instructions didn't mention any tabs on Amazon's site; they directed to an external site. So maybe this is a distractor.The third option says adding the item to the cart automatically applies the discount. That seems plausible because sometimes discounts are applied at checkout. If the external site lists items with existing discounts, adding them to the cart would reflect the discounted price. Also, the title mentions cheapsally.com as another site for discounts, which might be similar to tjshome.com. So following the steps from these sites would lead to items that, when added to the cart, have the discount applied automatically.So the most reasonable outcome is option 3. The other options either involve applying extra discounts (which isn't mentioned in the steps) or navigating tabs that aren't part of the original instructions.</think>The most reasonable scenario outcome is option 3. The steps describe using external sites to find discounted Amazon products. Adding these items to the cart would automatically apply the discounts listed on those sites, aligning with typical e-commerce behavior where discounts are reflected at checkout. Other options either introduce unmentioned steps (e.g., d̈iscounts tab̈) or imply altering advertised prices, which is less likely.boxed{answer3}

Deepseek-V3's Deep Thinking
The most reasonable scenario outcome based on the description of the young boy wearing a bandana and mowing a large yard, while being unrelieved by the weeds and barely smiling, is:**1. walks away from the camera as he pushes the mower.**This option aligns with the context of the boy focusing on the task of mowing the yard, without introducing unrelated activities like ping pong or talking to the camera.boxed{1}.

Gemini-2.0-preview's Deep Thinking

759

The most reasonable scenario outcome is: **2. combs and cuts the woman's hair as she sits.**The other options don't make logical sense in the context of a salon visit.boxed{answer}.

GLM-zero-preview's Deep Thinking
###Thinking Let's analyze the situation step by step. We have a child using his foot for the base and drumming on the snare with natural ease, while a woman is playing the piano alongside him. The phrase ẗheys̈uggests that both the child and the woman are actively involved in making music together.Option 1 mentions a small audience sitting and watching them perform. While it's possible that others are present, the original text doesn't provide any information about an audience. The focus is on the child and the woman playing music, not on any spectators. So, while this could be a scenario, it's not directly supported by the given text.Option 2 states that they continue to play the drum better and play till there's no longer a fist drumming loudly in the background. Wait, the original text doesn't mention anything about a fist drumming in the background. This option introduces new information that isn't present in the original description. It seems to be adding details that aren't there, so this option doesn't fit.Option 3 says they are both playing and singing along very intently while they're playing for joy. This option feels like a natural extension of the original text. We know they're playing music together, and it's reasonable to assume they might be singing along, especially given the joyful context implied by their natural and enthusiastic playing. The word ïntentlyf̈its with the description of the child drumming l̈ike a naturaländ the woman playing the piano alongside him. It's a scenario that aligns well with the original text.While option 1 is possible, it's not directly supported by the given information. Option 2 introduces new elements that aren't mentioned in the original text. Option 3, however, is a logical and reasonable outcome based on the description provided.Perhaps I'm overcomplicating this, but the original text sets a scene of music-making between the child and the woman. It feels natural to imagine them singing along as they play, fully engaged and enjoying themselves. The other options either introduce elements not present in the text or are less directly supported by the information given.After considering all the options, the most reasonable scenario outcome is that they are both playing and singing along very intently while they're playing for joy.**Answer:** [boxed{3}]###Response1. **Analyze the Given Text:**- A child is using his foot as the base and drumming on the snare naturally. - A woman is playing the piano alongside the child. - Both are actively engaged in making music together.2. **Evaluate Each Option:**- **Option 1:** Mentions a small audience watching the performance. - *Assessment:* The original text does not mention an audience. This scenario is possible but not directly supported.- **Option 2:** States they continue playing better until there's no longer a fist drumming loudly in the background.- *Assessment:* Introduces new information about a fist drumming, which is not present in the original text. This option does not fit.- **Option 3:** Describes both playing and singing intently for joy. - *Assessment:* Aligns well with the original text, as it is a natural extension of their music-making and engagement.3. **Determine the Most Reasonable Outcome:**- **Option 3** is the most logical and directly supported by the given information.**Final Answer:**[boxed{3}]

Deep Thinking. MATH's example is as follows:

MATH dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking
So I've got this problem here. Daniel works at an electronics store, and he has this theory that the popularity of a toaster, which is measured by the number of sales, is inversely proportional to its cost. So, if the cost goes up, the number of sales goes down, and vice versa.Okay, so mathematically, if something is inversely proportional, that means that one quantity is equal to a constant divided by the other quantity. So, in this case, the number of sales (let's call that n) is inversely proportional to the cost (let's call that c). So, n = k / c, where k is some constant.They give me some data to work with: 12 customers buy a toaster that costs $500. So, n = 12 when c = 500. I can use this to find

the constant k.Plugging in the values: 12 = k / 500. To find k, I can multiply both sides by 500: k = 12 * 500 = 6000.So, the relationship is n = 6000 / c.Now, they ask me how many customers would buy a toaster that costs $750. So, c = 750. Plugging that into the equation: n = 6000 / 750.Let me calculate that: 6000 divided by 750. Well, 750 times 8 is 6000. So, n = 8.So, according to Daniel's theory, 8 customers would buy a toaster that costs $750.Wait a minute, does that make sense? Let's double-check. Inverse proportion means that as c increases, n decreases, which is what happened here: from $500 to $750, n went from 12 to 8. That seems consistent.Is there another way to approach this problem? Maybe using ratios. In inverse proportion, the product of the two quantities is constant. So, n1 * c1 = n2 * c2.Given n1 = 12, c1 = 500, and c2 = 750, find n2.So, 12 * 500 = n2 * 750.That's 6000 = n2 * 750.Then, n2 = 6000 / 750 = 8.Same answer.Alright, I think that's correct.**Final Answer**[boxed{8}]

OpenAI-o1-preview's Deep Thinking
To solve this problem, we'll use the concept of inverse proportionality. If the number of sales ((S)) is inversely proportional to the cost ((C)), we can express this relationship as:[S = frac{k}{C}]where(k) is the constant of proportionality.First, we find the value of (k) using the information provided:When (C = $500), (S = 12):[12 = frac{k}{500} implies k = 12 times 500 = 6000]Now, we can find the number of sales when (C = $750):[S = frac{6000}{750} = 8] boxed{8}.

Deepseek-R1's Deep Thinking
<think>Okay, so I have this problem here where there's a set of four numbers, and the six pairwise sums of distinct elements are given as 189, 320, 287, 234, x, and y. I need to find the greatest possible value of x + y. Hmm, let's see. First, let me recall that if there are four numbers, say a, b, c, d, then the pairwise sums would be a+b, a+c, a+d, b+c, b+d, and c+d. Since there are six sums, that's correct. Now, these sums are given in no particular order, so I don't know which sum corresponds to which pair. The problem is asking for the maximum possible value of x + y, where x and y are the two missing sums.I remember that in problems like these, the sum of all the pairwise sums can be related to the sum of the original numbers. Let me think. If I add up all the pairwise sums, each number is added three times. For example, a is added to b, c, and d; similarly for the others. So, the total sum of all six pairwise sums is 3(a + b + c + d). So, if I add up all the given sums, including x and y, that should equal 3 times the total of the four numbers. Let me calculate the sum of the known values first: 189 + 320 + 287 + 234. Let me compute that step by step. 189 + 320 is 509. Then 509 + 287 is 796. Then 796 + 234 is 1030. So, the sum of the known four sums is 1030, and then we have x and y. So, the total sum of all six pairwise sums is 1030 + x + y = 3(a + b + c + d). Therefore, if I can find a + b + c + d, maybe I can relate it to x + y. But how? Well, since we need to maximize x + y, we need to minimize the sum of the other four sums. Wait, but the sum of all six sums is fixed once we know a + b + c + d. Wait, no. Wait, actually, the sum of all six sums is 3(a + b + c + d). So, if we can find a + b + c + d, then x + y would be 3(a + b + c + d) - 1030. Therefore, to maximize x + y, we need to maximize 3(a + b + c + d) - 1030. Which would mean maximizing a + b + c + d. But how do we find a + b + c + d? Hmm. Alternatively, maybe we can find a + b + c + d in terms of the given sums. Let me think. In problems with four numbers and their pairwise sums, sometimes arranging the sums in order can help. If we can order the sums from smallest to largest, then the smallest sum would be the sum of the two smallest numbers, and the largest sum would be the sum of the two largest numbers. The other sums would be in between. But here, the given sums are 189, 320, 287, 234, x, y. So, if we can figure out the order, maybe we can assign the known sums to positions in the sorted list. But since x and y are variables, we need to consider different possibilities. Alternatively, maybe there's a way to find the total sum a + b + c + d by using the given sums. Let me recall that if we have four numbers, the sum of all six pairwise sums is 3(a + b + c + d). So, if we can find the sum of all six pairwise sums, then we can divide by 3 to get a + b + c + d. But since two of the sums are missing (x and y), we need to figure out what their 763

possible values could be. But since we need to maximize x + y, perhaps we can find the maximum possible value of x + y such that all six sums are possible with some four numbers a, b, c, d. Wait, but how do we ensure that such numbers a, b, c, d exist? That's the key. The sums must be consistent with some set of four numbers. So, even if we try to maximize x + y, we need to make sure that the resulting set of sums can be formed by some four numbers. Let me think. Let's denote the four numbers in increasing order as a <= b <= c <= d. Then the pairwise sums would be:a+b (smallest sum).

Deepseek-V3's Deep Thinking
To determine the probability that the selected ball has a prime number, follow these steps:1. **Identify the total number of balls:** There are 6 balls numbered 2, 3, 4, 5, 6, and 7.2. **Identify the prime numbers among the balls:**The prime numbers in the list are 2, 3, 5, and 7. So, there are 4 prime numbers.3. **Calculate the probability:** Probability is given by the ratio of the number of favorable outcomes to the total number of possible outcomes.[text{Probability} = frac{text{Number of prime-numbered balls}}{text{Total number of balls}} = frac{4}{6}]4. **Simplify the fraction:**[frac{4}{6} = frac{2}{3}]Therefore, the probability that the selected ball has a prime number is:[boxed{dfrac{2}{3}}].

Gemini-2.0-preview's Deep Thinking
Let $x = 0.overline\{9\}$. Then $x = 0.9999dots$.Multiplying by 10, we get $10x = 9.9999dots = 9.overline\{9\}$.Subtracting x from $10x$, we have $10x - x = 9.overline\{9\} - 0.overline9$, which simplifies to $9x = 9$.Dividing by 9, we get $x = 1$.Therefore, $0.overline\{9\} = 1$.Then $1 - 0.overline\{9\} = 1 - 1 = 0$.Thus, $1 - 0.overline\{9\} = 0$.Final Answer: The final answer is $boxed\{0\}$.

GLM-zero-preview's Deep Thinking
###ThinkingLet's tackle this problem step by step to determine how many three-digit codes Reckha can choose.My code is 023. Reckha's code cannot be the same as mine in two or more digit positions, nor can it be the same as mine except for switching the positions of two digits. So, first, let's identify all the forbidden codes.**Forbidden Codes Due to Switching Two Digits:**We can switch the digits of 023 in three ways:- Switch the first and second digits: **203**- Switch the first and third digits: **320**- Switch the second and third digits: **032**So, the codes 203, 320, and 032 are forbidden.Wait, is that all? Let's make sure we're not missing any permutations. The number 023 has digits 0, 2, and 3. The total permutations of these digits are:- 023- 032- 203- 230- 302- 320Ah, we've missed 230 and 302! But are these forbidden? The problem specifically mentions codes that are the same as mine except for switching the positions of two digits.=Let's think about what s̈witching the positions of two digitsm̈eans. Switching two digits involves swapping the positions of any two digits in the original code.From 023:- Switching the first and second digits: **203**- Switching the first and third digits: **320**- Switching the second and third digits: **032**These are the only codes obtained by switching exactly two digits. Codes like 230 and 302 involve more than just switching two digits they are permutations that require more than one swap.For example, to get from 023 to 230:- Swap the first and second digits to get 203.- Then swap the second and third digits to get 230.This involves two swaps, not just one. Similarly, to get 302:- Swap the first and third digits to get 320.- Then swap the second and third digits to get 302.Again, two swaps. So, 230 and 302 are not obtained by switching just two digits they are forbidden only if they fall under the other restriction.Wait, perhaps we should consider whether the problem means codes that are the same as mine except for switching any two digits (which could imply any permutation obtained by swapping any two digits, possibly multiple times), but the wording s̈witching the positions of two digitsẗypically means a single swap.However, to be thorough, let's consider both interpretations.First, let's stick with the strict interpretation: switching exactly two digits.So, the forbidden codes from switching two digits are:- 203 (swap first and second digits)- 320 (swap first and third digits)- 032

(swap second and third digits)Now, let's consider the codes that are the same as mine in two or more digit positions.**Forbidden Codes Due to Matching Two or More Digits in the Same Positions:**We need to find all codes that match 023 in exactly two positions or all three positions.Let's break this down:1. **Codes that match in all three positions:**- **023**2. **Codes that match in exactly two positions:** - **First and second digits match (0, 2, _):**- The first digit is 0. - The second digit is 2. - The third digit can be any digit except 3 (to avoid matching all three digits). So, the codes are 020, 021, 022, 024, 025, 026, 027, 028, 029. **First and third digits match (0, _, 3):**- The first digit is 0.- The second digit can be any digit except 2.- The third digit is 3. Codes: 003, 013, 033, 043, 053, 063, 073, 083, 093.- **Second and third digits match (_, 2, 3):** - The first digit can be any digit except 0. - The second digit is 2.

Deep Thinking. MBPP's example is as follows:

MBPP dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking
Alright, I have this problem to solve: Ẅrite a Python function to find the sum of the fourth power of the first n even natural numbers.Ökay, let's break this down.First, I need to understand what are even natural numbers. Natural numbers are like 1, 2, 3, 4, and so on. Even natural numbers would be 2, 4, 6, 8, etc.So, the first n even natural numbers would be the first n numbers in this sequence: 2, 4, 6, 8, ... Now, I need to find the fourth power of each of these numbers and then sum them up.Let me try with a small n to understand the pattern. Let's say n=3.So, the first 3 even natural numbers are 2, 4, and 6.Their fourth powers would be:2^4 = 16 4^4 = 256 6^4 = 1296 Sum of these is 16 + 256 + 1296 = 1568.Okay, now I need a general formula or a way to calculate this for any n.One way is to iterate through the first n even numbers, calculate their fourth power, and sum them up. But since this is Python, maybe there's a more efficient way.Let me think about the formula for the sum of fourth powers of the first n even natural numbers.An even number can be represented as 2k, where k is a natural number.So, the first n even natural numbers are 2*1, 2*2, 2*3, ..., 2*n.Their fourth powers would be (2*1)^4, (2*2)^4, ..., (2*n)^4.Which simplifies to 16*(1^4), 16*(2^4), ..., 16*(n^4).So, the sum would be 16*(1^4 + 2^4 + 3^4 + ... + n^4).Now, I need the formula for the sum of fourth powers of the first n natural numbers.The formula for the sum of the fourth powers is:sum = n*(n+1)*(2*n+1)*(3*n^2 + 3*n - 1)/30 You can find this formula in mathematical references.So, using this, the sum of the fourth powers of the first n even natural numbers would be: 16 * [n*(n+1)*(2*n+1)*(3*n^2 + 3*n - 1)/30]Simplifying this: (16/30)*n*(n+1)*(2*n+1)*(3*n^2 + 3*n - 1)Which can be written as: (8/15)*n*(n+1)*(2*n+1)*(3*n^2 + 3*n - 1)But maybe it's better to keep it in the original form for the code.Now, I need to implement this in Python.I can write a function that takes n as input and returns this sum.I should also ensure that n is a positive integer, perhaps handle cases where n is not positive by returning an error or zero.Let me also verify this formula with the earlier example where n=3.sum = 16*(1^4 + 2^4 + 3^4) = 16*(1 + 16 + 81) = 16*98 = 1568.Using the formula:sum = 16 * [n*(n+1)*(2*n+1)*(3*n^2 + 3*n - 1)/30] Plugging n=3: sum = 16*[3*4*7*(27 + 9 -1)/30] = 16*[12*7*35/30] = 16*[12*7*35 / 30]Calculate inside the brackets:12*7 = 84 84*35 = 2940 2940 / 30 = 98 So, 16*98 = 1568, which matches the earlier calculation.Good, the formula seems correct.Now, thinking about implementing this in Python.I can define a function, say, sum_of_fourth_powers_even(n):# Calculate sum of fourth powers of first n even natural numbers return 16 * sum(k**4 for k in range(1, n+1)) But this is straightforward but not using the formula.

OpenAI-o1-preview's Deep Thinking
def find_cumulative_sum(tuple_list): total_sum = 0 for tup in tuple_list: total_sum += sum(tup) return total_sum # Example usage: tuple_list = [(1, 2), (3, 4), (5, 6)] answer = find_cumulative_sum(tuple_list) boxed{answer}.

Deepseek-R1's Deep Thinking

<think>Okay, I need to write a Python function that calculates the sum of the fourth powers of the first n even natural numbers. Let me think about how to approach this.First, let's clarify the problem. The first n even natural numbers are 2, 4, 6, ..., 2n. So for each of these numbers, I have to compute the fourth power and then sum them all up.Wait, wait. Wait, the first n even numbers. Let me confirm: if n is 3, then the numbers are 2, 4, 6. Their fourth powers would be 16, 256, 1296. Sum is 16 + 256 + 1296 = 1568. So the function should return that when n=3.So the general approach is to generate each even number up to the nth term, compute the fourth power, and accumulate the sum.But maybe there's a formula to compute this without looping? That might be more efficient, especially for large n. Let me think about mathematical formulas.The sum of the fourth powers of the first m even numbers. Let's express the even numbers as 2k where k ranges from 1 to m. So the sum is sum_{k=1 to m} (2k)^4.Expanding (2k)^4: 16k^4. So the sum becomes 16 * sum_{k=1 to m} k^4.Now, the sum of the fourth powers of the first m natural numbers is a known formula. Let me recall that formula.The formula for sum_{k=1 to m} k^4 is m(m + 1)(2m + 1)(3m**2 + 3m 1)/30. Let me check that. Hmm, I think that's correct. Let me verify for m=1: 1*2*3*(3 +3 -1)/30 = 1*2*3*5 /30 = 30/30=1. Correct. For m=2: 2*3*5*(12 +6 -1)/30. Wait, 3m is 12, 3m is 6, so 12+6-1=17. So 2*3*5*17 /30 = (30*17)/30=17. Sum of 1^4 +2^4=1+16=17. Correct. So the formula works.So substituting back, the sum of the fourth powers of the first m even numbers is 16 * [m(m + 1)(2m + 1)(3m**2 + 3m 1)/30].But wait, in our problem, the first n even numbers. Wait, in the problem statement, the first n even natural numbers are 2,4,...,2n. So m here is n. So the sum is 16 * sum_k=1 to n k^4. So substituting the formula, we can compute it directly.Therefore, the function can be written using this formula. Alternatively, we can loop through each even number, compute the fourth power, and add them up. Which approach is better? For code clarity and efficiency, using the formula is better, especially for large n. But for small n, either approach works.But the problem says to write a Python function. Let's consider both approaches.First, the loop approach. Let's say n is given. We can initialize a sum variable to 0. Then loop from 1 to n, take 2*i as the even number, compute (2*i)**4, add to the sum. Then return the sum.Alternatively, using the formula. Let's code the formula. Let's compute m = n. Then compute 16 * m*(m+1)*(2m+1)*(3m**2 +3m -1)/30.But wait, let's check for n=3. Using the formula: m=3. 3*4=12, 2*3+1=7, 3*(3)^2 +3*3 -1= 27 +9 -1=35. So 12*7=84, 84*35=2940. Then 2940 *3= 8820? Wait, wait. Wait the formula is m(m+1)(2m+1)(3m**2 +3m -1)/30. So 3*4*7*35 /30. Let's compute that: 3*4=12, 12*7=84, 84*35=2940. 2940 /30=98. Then multiply by 16: 98*16=1568. Which matches the example I had earlier.

Deepseek-V3's Deep Thinking

To perform the modulo operation on corresponding elements of two tuples, we can write a Python function that takes two tuples as input and returns a new tuple containing the result of the modulo operation for each pair of elements. Here's how you can do it: def modulo_tuples(tuple1, tuple2):# Ensure both tuples have the same length if len(tuple1) != len(tuple2): raise ValueError(B̈oth tuples must have the same length)̈# Perform modulo operation on corresponding elements result = tuple(a % b for a, b in zip(tuple1, tuple2)) return result# Example usage: tuple1 = (10, 20, 30) tuple2 = (3, 7, 4) result = modulo_tuples(tuple1, tuple2) print(result) # Output will be (1, 6, 2)### Explanation: 1. **Input Validation**: The function first checks if the two tuples have the same length. If not, it raises a 'ValueError'.2. **Modulo Operation**: It then uses a generator expression within a 'tuple' constructor to perform the modulo operation ('%') on corresponding elements of the two tuples.3. **Return Result**: The function returns the resulting tuple.### Final Answer:boxed{text{The function 'modulo_tuples' performs the modulo operation on corresponding elements of two tuples and returns the result as a new tuple.}}.

Gemini-2.0-preview's Deep Thinking

def cumulative_sum_tuple_list(tuple_list): Calculates the cumulative sum of all values in a list of tuples.Args: tuple_list: A list of tuples containing numerical values. Returns:The cumulative sum of all values in the tuple list. cumulative_sum = 0 for tup in tuple_list: for value in tup: cumulative_sum += value return cumulative_sum# Example usage (not required for the solution, but good for testing)# my_list = [(1, 2), (3, 4), (5, 6)]# result = cumulative_sum_tuple_list(my_list)# print(result) # Output: 21# Store the function definition in the answer variable answer = cumulative_sum_tuple_list.

GLM-zero-preview's Deep Thinking

###ThinkingLet's tackle this problem of exchanging the positions of every n-th value with the (n+1)th value in a given list. At first glance, it seems straightforward, but let's make sure we understand exactly what's being asked.We need to swap every n-th element with the (n+1)th element. For example, if ‘n = 2‘, we should swap the 2nd and 3rd elements, then the 4th and 5th elements, and so on. If ‘n = 3‘, we swap the 3rd and 4th elements, then the 6th and 7th elements, and so forth.Wait, let's make sure we're interpreting the indexing correctly. In programming, lists are typically zero-indexed, meaning the first element is at position 0, the second at position 1, and so on. However, the problem refers to the n̈-thv̈alue, which is one-indexed. So, we need to adjust our indices accordingly.for ‘n = 2‘, the 2nd element is at index ‘1‘, and the 3rd element is at index ‘2‘. For ‘n = 3‘, the 3rd element is at index ‘2‘, and the 4th element is at index ‘3‘. So, in general, the n-th element is at index ‘n-1‘, and the (n+1)th element is at index ‘n‘.Now, let's think about how to implement this. We can iterate over the list in steps of ‘n + 1‘, so for each iteration, we'll be at the starting index for the swap. Let's consider ‘n = 2‘ as an example. Our list is ‘[1, 2, 3, 4, 5, 6, 7]‘. We want to swap elements at indices ‘1‘ and ‘2‘, then ‘3‘ and ‘4‘, and so on.If we use a range with ‘range(n - 1, len(lst) - 1, n + 1)‘, let's see what happens:- For ‘n = 2‘, ‘n - 1 = 1‘.- The range becomes ‘range(1, 6, 3)‘, which gives indices ‘1‘, ‘4‘.- At index ‘1‘, we swap with index ‘2‘.- At index ‘4‘, we swap with index ‘5‘.Wait, that seems to skip the last element if the list length isn't a multiple of ‘n + 1‘. Let's test it with our list ‘[1, 2, 3, 4, 5, 6, 7]‘:- Swap indices ‘1‘ and ‘2‘: ‘[1, 3, 2, 4, 5, 6, 7]‘- Swap indices ‘4‘ and ‘5‘: ‘[1, 3, 2, 4, 6, 5, 7]‘The last element ‘7‘ remains unchanged, which is fine.But what if ‘n = 3‘ and the list is ‘[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]‘?- ‘n - 1 = 2‘- The range becomes ‘range(2, 9, 4)‘, which gives indices ‘2‘, ‘6‘.- Swap indices ‘2‘ and ‘3‘: ‘[0, 1, 3, 2, 4, 5, 6, 7, 8, 9]‘- Swap indices ‘6‘ and ‘7‘: ‘[0, 1, 3, 2, 4, 5, 7, 6, 8, 9]‘Wait, we're missing the swap between indices ‘10‘ and ‘11‘ because our range goes up to ‘len(lst) - 1‘, which is ‘9‘. Index ‘9‘ corresponds to the 10th element, but we need to swap index ‘9‘ with index ‘10‘, which doesn't exist. So, we need to adjust our range to ensure we don't go out of bounds.Perhaps we should use ‘range(n - 1, len(lst) - 1, n + 1)‘. Let's try that:For ‘n = 3‘, ‘n - 1 = 2‘, ‘len(lst) - 1 = 9‘, so the range is ‘range(2, 9, 4)‘, giving indices ‘2‘, ‘6‘.- Swap indices ‘2‘ and ‘3‘: ‘[0, 1, 3, 2, 4]

Deep Thinking. HumanEval's example is as follows:

HumanEval dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking

So I've got this problem to solve. I need to figure out if a given number ‘n‘ can be written as the sum of exactly four positive even numbers. Let's break this down.First, what's a positive even number? It's a number like 2, 4, 6, 8, and so on. So, I need to find four of these numbers that add up to ‘n‘.The examples given are:- ‘is_equal_to_sum_even(4)‘ should return ‘False‘.- ‘is_equal_to_sum_even(6)‘ should return ‘False‘.- ‘is_equal_to_sum_even(8)‘ should return ‘True‘.Let's think about why these are the cases.Starting with ‘n = 4‘:What's the smallest sum of four positive even numbers? The smallest positive even numbers are 2, 2, 2, and 2, which sum up to 8. So, 4 is less than the smallest possible sum, which is 8. That's why ‘is_equal_to_sum_even(4)‘ is ‘False‘.Similarly, for ‘n = 6‘, which is still less than 8, it's impossible to get 6 as the sum of four positive even numbers. Hence, ‘False‘.For ‘n = 8‘, it's possible with 2 + 2 + 2 + 2 = 8. So, ‘True‘.Okay, so it seems like any number greater than or equal to 8 can be expressed as the sum of four positive even numbers. But wait, is that always true?Let's consider 'n = 10':2 + 2 + 2 + 4 = 10. That works.'n = 12':2 + 2 + 4 + 4 = 12. Also works.'n = 14':2 + 4 + 4 + 4 = 14. Still works.'n = 16':4 + 4 + 4 + 4 = 16. Good.What about 'n = 9'? That's an odd number. But the problem specifies positive even numbers, so the sum would always be even. Hence, any odd 'n' can't be expressed as the sum of four even numbers. So, 'False' for odd 'n'.Wait, but the problem says p̈ositive even numbers,̧ so they have to be even and positive. So, no zeros or negatives.So, from this, I can generalize that for any even 'n' greater than or equal to 8, it's possible, and for any odd 'n', it's not possible.But let's test this with another even number greater than 8.'n = 18':4 + 4 + 5 + 5 = 18, but 5 is not even. So, 4 + 4 + 4 + 6 = 18. That works.'n = 20':4 + 4 + 6 + 6 = 20. Good.'n = 24':6 + 6 + 6 + 6 = 24. Perfect.Now, what about 'n = 8' itself, which we already know is True.And 'n = 4' and '6' are less than 8 and hence False.So, it seems like the rule is: if 'n' is even and 'n >= 8', then True; else, False.But wait, let's think about 'n = 8'. It's the smallest possible sum with four even numbers, all being 2.Is there any even number greater than or equal to 8 that cannot be expressed as the sum of four even numbers? Let's see 'n = 12': Option 1: 2 + 2 + 2 + 6 = 12 Option 2: 2 + 2 + 4 + 4 = 12 Option 3: 2 + 3 + 3 + 4 = 12, but 3 is odd, so invalid.So, only even numbers, positive, summing to 12.Seems like it's always possible for even 'n' >= 8.Let's consider 'n = 2'.Smallest sum is 8, so False.'n = 0', but the problem specifies positive even numbers, so 'n' should be positive.'n = 1', which is odd, so False.'n = 10': as above, True.So, the general rule is: if 'n' is even and 'n >= 8', return True.

OpenAI-o1-preview's Deep Thinking
def can_arrange(arr): ans = -1 for i in range(1, len(arr)): if arr[i] < arr[i-1]: ans = i return ans.

Deepseek-R1's Deep Thinking
<think>Okay, I need to write a Python function called fib that returns the n-th Fibonacci number. Let's see, the examples given are fib(10) should return 55, fib(1) is 1, and fib(8) is 21. Hmm, wait, the Fibonacci sequence usually starts with 0 and 1, right? But in the examples, fib(1) is 1. Let me check. Oh, right, sometimes the sequence is indexed starting from 1 where fib(1) = 1, fib(2) = 1, fib(3) = 2, etc. So for n=10, the 10th number would be 55. Let me confirm that.Let me list out the Fibonacci numbers with that indexing. Let's see:n: 1 2 3 4 5 6 7 8 9 10 fib(n): 1,1,2,3,5,8,13,21,34,55. Yes, that's correct. So fib(8) is 21, which matches the example. So the function needs to return the nth number in this sequence.Now, how to implement this. The straightforward recursive approach would be too slow for large n, but since the problem doesn't specify constraints on n's size, maybe a simple iterative approach would work. Let's think about the iterative method.Initialize two variables, a and b, as the first two numbers. Then, for each step up to n, compute the next number. Wait, but for n=1, we return 1. Let's see:If n is 1 or 2, return 1. For n > 2, iterate from 3 to n, each time adding the previous two.So the steps would be:def fib(n): if n == 1 or n == 2: return 1 a, b = 1, 1 for _ in range(3, n+1):c = a + b a = b b = c return b Wait, let's test this with the examples.For n=1: returns 1. Correct.n=8: Let's compute step by step.Start with a=1, b=1. Then loop from 3 to 8 (6 iterations):3: c=2, a=1, b=2 4: c=1+2=3, a=2, b=35: c=2+3=5, a=3, b=56: c=3+5=8, a=5, b=8 7: c=5+8=13, a=8, b=138: c=8+13=21, a=13, b=21. Then return b=21. Correct.n=10: Let's see. After 8 steps, we have a=13, b=21. Then for 9th step (n=9):c=13+21=34, a=21, b=34. Then for n=10, next iteration:c=21+34=55, a=34, b=55. Return 55. Correct.So this code should work. But wait, what if n is 0? The problem says n is an integer, but the examples start at 1. The docstring says n̈-th Fibonacci number,̧ and the examples include n=1. So perhaps the function is intended to handle n >=1. So the code as written should be okay.Alternatively, maybe handle n=0? But the problem's examples don't include that, and the docstring's first example is n=10. So probably, the code is correct as written.Another approach could be using recursion with memoization, but for larger n, recursion would hit stack limits. The iterative approach is better here.So the code should be as I wrote above. Let me check the code again.Yes, the code checks if n is 1 or 2, returns 1. Then loops from 3 to n,

772

updating a and b each time. Finally returns b.So that's the solution.</think>To solve this problem, we need to compute the n-th Fibonacci number, where the sequence starts with 1 for both the first and second positions. The Fibonacci sequence is defined such that each subsequent number is the sum of the two preceding ones.### ApproachThe approach used here is an iterative method to efficiently compute the Fibonacci number. This method avoids the exponential time complexity of the naive recursive approach by using constant space and linear time. The key steps are:1. Check if the input 'n' is 1 or 2.

Deepseek-V3's Deep Thinking
To solve the problem of extracting only the positive numbers from a given list, we can iterate through the list and filter out the elements that are greater than 0. Here's how we can implement the 'get_positive' function def get_positive(l: list): Return only positive numbers in the list. return [x for x in l if x > 0]### Explanation:- We use a list comprehension to iterate through each element 'x' in the list 'l'.- We include 'x' in the new list only if 'x > 0'.### Example Usage:"'python print(get_positive([-1, 2, -4, 5, 6])) # Output: [2, 5, 6] print(get_positive([5, 3, -5, 2, -3, 3, 9, 0, 123, 1, -10])) # Output: [5, 3, 2, 3, 9, 123, 1].

Gemini-2.0-preview's Deep Thinking
def rounded_avg(n, m): You are given two positive integers n and m, and your task is to compute the average of the integers from n through m (including n and m). Round the answer to the nearest integer and convert that to that to binary. If n is greater than m, return -1.Example: rounded_avg(1, 5) => Öb11ŕounded_avg(7, 5) => -1 rounded_avg(10, 20) => Öb1111ŕounded_avg(20, 33) => Öb11010ïf n > m: return -1 total = 0 for i in range(n, m + 1): total += i avg = round(total / (m - n + 1)) return bin(avg).

GLM-zero-preview's Deep Thinking
###Thinking Let's tackle this problem of determining whether a given number 'n' can be written as the sum of exactly four positive even numbers. The first thing that comes to mind is understanding what positive even numbers are: they're numbers like 2, 4, 6, 8, and so on. So, we're looking for four such numbers that add up to 'n'.Let's start by considering the smallest possible sum of four positive even numbers. The smallest even number is 2, so the smallest sum we can get is:[2 + 2 + 2 + 2 = 8]This means that any number less than 8 cannot be expressed as the sum of four positive even numbers. So, right off the bat, we know that 'is_equal_to_sum_even(n)' will be 'False' for 'n < 8'.Wait, what about 'n = 8'? Let's check:[2 + 2 + 2 + 2 = 8]Yes, that works! So, 'is_equal_to_sum_even(8)' should return 'True'.Now, let's consider 'n = 9'. Since 9 is odd, and the sum of four even numbers is always even (because even + even = even), 9 cannot be expressed as the sum of four positive even numbers. Therefore, 'is_equal_to_sum_even(9)' should be 'False'.But what about even numbers greater than 8? Let's try 'n = 10':[2 + 2 + 2 + 4 = 10]Perfect! And for 'n = 12':[2 + 2 + 4 + 4 = 12]Or even:[2 + 4 + 2 + 4 = 12]Wait, perhaps we can find a pattern here. Let's consider the general case for any even number 'n >= 8'.Let's express 'n' as:[n = 2 + 2 + 2 + (n - 6)]For this to work, '(n - 6)' must be a positive even number. Let's check:- **Is '(n - 6)' positive?**Since 'n >= 8', we have:[n - 6 >= 8 - 6 = 2] So, '(n - 6)' is at least 2, which is positive. - **Is '(n - 6)' even?** Since 'n' is even, let's let 'n = 2k' for some integer 'k'. Then:[n - 6 = 2k - 6 = 2(k - 3)] This shows that '(n - 6)' is even because it's a multiple of 2.So, for any even 'n >= 8', we can express it as:[n = 2 + 2 + 2 + (n - 6)]where '(n - 6)' is a positive even number. This means that any even number greater than or equal to 8 can be written as the sum of exactly four positive even numbers.However, what about odd numbers greater than 8? Let's consider 'n = 11'. If we try to express 11 as the sum of four positive even numbers, we run into a problem because the sum of four even numbers is always even, and 11 is odd. So, it's impossible.Wait, let's test this with 'n = 10' (even) and 'n = 11' (odd) to be sure.For 'n = 10':- (2 + 2 + 2 + 4 = 10)- (2 + 2 + 3 + 3 = 10) but 3 is odd, so that doesn't work.- (2 + 4 + 2 + 2 = 10)All valid combinations use even numbers, and the sum is even.For 'n = 11':- (2 + 2 + 2 + 5 = 11

) but 5 is odd.- (2 + 2 + 3 + 4 = 11) again, 3 is odd.- (2 + 4 + 2 + 3 = 11) still, 3 is odd.No matter what, we can't find four positive even numbers that add up to 11. So, our initial thought holds: the sum of four positive even numbers is always even.

Deep Thinking SST-2's example is as follows:

SST-2 dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking
So I have this sentence: g̈ets under your skin and, some plot blips. If you think this sentence is positive, the answer is 1, and the negative answer is 0.F̈irst, I need to understand what the sentence is saying. G̈ets under your skinïs an idiom that usually means something is annoying or bothersome, or it could also mean something is deeply affecting or engaging. S̈ome plot blipss̈uggests there are minor issues or glitches in the plot.So, overall, the sentence seems to be pointing out negative aspects: something being annoying or having plot issues. Therefore, if I'm to determine whether this sentence is positive or negative, it seems negative.Given that, if the sentence is positive, answer is 1; if negative, answer is 0. Since I've determined it's negative, the answer should be 0.But let's double-check. Maybe g̈ets under your skinc̈an be positive in some contexts, like something being very engaging or memorable. However, in this phrase, it's paired with s̈ome plot blips,ẅhich are definitely negative. So, overall, it's likely negative.

OpenAI-o1-preview's Deep Thinking
boxed{0}.

Deepseek-R1's Deep Thinking
<think>Okay, let's see. The user wants me to determine if the sentence g̈ets under your skin and , some plot blipsïs positive or negative. The answer should be 1 for positive and 0 for negative.First, I need to analyze the phrases here. G̈ets under your skinïs an idiom. I remember that this can have different meanings depending on context. Usually, it means something that's irritating or bothersome. Like when something annoys you, it gets under your skin. But sometimes it can also mean something that affects you emotionally in a deeper way, maybe even in a positive sense if it's really moving. But I think the more common usage is negative, like being annoyed.Then there's s̈ome plot blips.̈ B̈lipsḧere probably refers to minor issues or problems in the plot. A blip is a small, temporary problem or a minor error. So s̈ome plot blipsẅould mean there are a few small issues with the story. That sounds like a negative point, pointing out flaws in the plot.Putting them together: The first part could be either, but leaning negative. The second part is definitely pointing out negatives. So the overall sentiment is likely negative..

Deepseek-V3's Deep Thinking
The sentence ẗell the best storyïs a request or an encouragement to share a story, which is generally positive in tone. It does not convey negativity..

Gemini-2.0-preview's Deep Thinking
The sentence ïmagination and insightëvokes positive qualities..

GLM-zero-preview's Deep Thinking
###ThinkingLet's analyze the sentence: T̈he end result is like cold porridge with only the odd enjoyably chewy lump.F̈irst, let's break it down. The comparison to c̈old porridgeïmmediately evokes a negative image. Cold porridge is often unappetizing stale, unpalatable, and lacking in appeal. It's not a desirable state for food, suggesting something that's unenjoyable or unsatisfactory.Wait, but there's a twist with önly the odd enjoyably chewy lump.T̈his phrase introduces a slightly positive

element a rare, enjoyable aspect within an otherwise unappealing context. The ënjoyably chewy lumpïs a pleasant surprise, a moment of texture that stands out. However, the use of önly the oddëmphasizes that these enjoyable bits are few and far between, almost negligible compared to the overall experience.So, we have a predominantly negative description (c̈old porridge)̈ with a tiny positive (ẗhe odd enjoyably chewy lump)̈. The negative imagery sets the tone, and the positive element is too sparse to outweigh it. It's like saying a meal is tasteless and bland with just a single tasty bite overall, it's still a disappointing experience.Initially, I might have been tempted to consider the positive aspect more heavily, thinking that the ënjoyably chewy lumpc̈ould tip the balance. Maybe the chewy bits are so delightful that they make up for the cold porridge. But then I realized that the phrase önly the oddr̈eally diminishes their impact. It's not a bowl full of chewy lumps; it's just an occasional pleasant surprise in a sea of mediocrity or worse.Let's think about it another way. If someone described their day as Ïike trudging through mud with only the odd sunny spot,ÿou'd probably conclude it was a bad day overall, right? The sunny spots are nice, but they don't negate the misery of trudging through mud. Similarly, in this sentence, the cold porridge dominates the experience.I also considered the potential for sarcasm or irony. Could the speaker be being sarcastic, making the cold porridge sound worse than it is? But without additional context, it's hard to justify that interpretation. The sentence seems straightforward, and the negative imagery is quite strong.

Deep Thinking and Prompts. IMDB's example is as follows:

IMDB dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking
So I've got this task here. There's a text, which is a review about a movie, and the question is: Ïf you think this sentence is positive, the answer is 1, and the negative answer is 0.T̈he instruction is to read the text, determine if it's positive or negative, and then provide either a 1 or a 0 based on that judgment.First, I need to understand what the text is saying. It's a movie review, and from the beginning, it seems like the person really didn't enjoy the movie at all. They call it a ḧorrible movieänd say they have only themselves to be angry with for watching it. They mention that they loved another movie called Ẅar Games,b̈ut this one was a disappointment.The reviewer expresses regret for watching the movie and even says they'll live to regret that decision for the rest of their life, which sounds very dramatic and negative. They also criticize the plot, dialogue, and acting, calling them very bad. There's even a mention of a s̈teaming pile of horse dung,ẅhich is a pretty strong negative expression.

OpenAI-o1-preview's Deep Thinking
boxed{0}.

Deepseek-R1's Deep Thinking
<think>Okay, let's see. The user provided a movie review and wants to know if it's positive (1) or negative (0). The task is to analyze the text and determine the sentiment.First, I'll read through the review again. The first part says S̈loppily directed, witless comedy...T̈hat's already negative. Words like s̈loppilyänd ẅitlessäre strong negatives. Then it mentions that the movie is supposed to spoof 50s alien invasion films but is no better than them, except for technical aspects like makeup effects. Even though there's a mention of good makeup, the overall comparison is negative because the spoof is worse than the originals. The user also references other failed spoofs like C̈asino Royaleänd Öur Man Flint,s̈aying they're worse than Bond movies, implying this movie is similarly bad. Finally, the reviewer says after two hours of screeching voices, you'll want peace and quiet. That's another negative point about the experience.

Deepseek-V3's Deep Thinking

The sentence describes the movie as s̈loppily directed,ẅitless,änd n̈o better thanẗhe films it spoofs, except for technical aspects like makeup effects. It also mentions that the spoof is ẅorse than its targetänd that the experience of watching it is unpleasant. These are all negative critiques.

Gemini-2.0-preview's Deep Thinking
The text is overwhelmingly negative. Here's a breakdown:* S̈loppily directed, witless comedy- Negative* s̈upposedly spoofs...but really is no better than them- Negative* äny spoof that is worse than its target is doomed to fail- Negative* Examples given (C̈asino Royale, Öur Man Flint) are presented as failures.* Äfter two hours of hearing the screeching voices of the aliens, you'll be begging for some peace and quiet- Very negative.* Rating of (*1/2) is extremely low..

GLM-zero-preview's Deep Thinking
###ThinkingLet's tackle this movie review and determine whether it's positive or negative. The review mentions that it's the ẗouching story of two families,ẅhich immediately sets a warm and emotional tone. The fact that it's about ẗhe relationships within each familys̈uggests a focus on the human connections that are often at the heart of compelling dramas.The review states that ëach family has a gay son,änd ẗhe stories are interrelated at that point.T̈his indicates that the shared experience of having gay sons is a central theme, but it's not the only focus. Importantly, the film is äbout all of the family members, not just the two sons.T̈his broad scope suggests a well-rounded exploration of the families' dynamics.We're told that ẗhe portraits of each of the family members in both families are well drawn.T̈his is a strong compliment, implying that the characters are complex and realistically portrayed. The reviewer also mentions that ẗhe story is consistently interesting,ẅhich is a clear positive. It keeps the audience engaged throughout.However, the review does include the phrase ä bit bleak.T̈his is a negative element, suggesting that the tone of the film is somber or depressing. But wait, is this enough to tip the review into the negative category?Let's consider the context. The bleakness might be inherent to the subject matter, and the reviewer doesn't seem to criticize the film for being bleak but rather states it as a fact. The reviewer doesn't elaborate on the bleakness or suggest that it detracts from the overall quality of the film.Thinking about it, the review starts with positive aspects the touching story, the focus on relationships, the well-drawn portraits and only mentions the bleakness towards the end. It's possible that the reviewer sees the bleakness as a necessary part of the story rather than a flaw.Perhaps the reviewer is saying that despite the bleakness, the film is consistently interesting. If the film were overly bleak or depressing, we might expect a stronger negative reaction. The use of ä bitälso softens the impact of the criticism.Let's also consider the overall tone of the review. It feels more positive than negative. The reviewer highlights several strengths of the film without mentioning any significant flaws beyond the bleakness, which seems to be part of the film's design.I did initially think the mention of b̈leakm̈ight make this a negative review, but perhaps I'm overcomplicating it. The review seems to be saying the movie is good, with well-drawn characters and an interesting story, even if it's a bit depressing.Let's not overthink this. The review is more positive than negative. The b̈it bleakïs outweighed by the positives.

Deep Thinking SQUAD's example is as follows:

SQUAD dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking
So I've got this question here: Ïn what borough is Fort Hamilton located?änd the text provides some information about Fort Hamilton and other military installations in New York City. Let's see what it says about Fort Hamilton.The text mentions that Fort Hamilton is ẗhe U.S. military's only active duty installation within the city.Ït goes on to say that it was ëstablished in 1825 in Brooklyn on the site of a small battery utilized during the American Revolution.S̈o, right here, it specifies that Fort Hamilton is in Brooklyn.Just to make sure, I should confirm if Brooklyn is a borough of New York

City. Yes, I know that New York City has five boroughs: Manhattan, Brooklyn, Queens, the Bronx, and Staten Island. So, Brooklyn is indeed one of the boroughs.The text also provides some additional information about Fort Hamilton, like its current functions, but the key point for this question is that it's located in Brooklyn.

Deepseek-R1's Deep Thinking
<think>Okay, let's see. The question is asking about an event that Queen Victoria attended in 1866 for the first time since Albert's death. I need to find the answer in the provided text.Looking at the text, the first sentence mentions that Palmerston died in 1865, then Russell had a brief ministry, and Derby returned to power. Then it says, Ïn 1866, Victoria attended the State Opening of Parliament for the first time since Albert's death.T̈hat seems to directly answer the question. The event is the State Opening of Parliament. Wait, let me double-check. The next sentences talk about 1867 and the Reform Act, but the key part is the 1866 event. The text clearly states that in 1866 she attended the State Opening of Parliament for the first time since Albert's death. Albert died in 1861, so this would be five years later.

Deepseek-V3's Deep Thinking
Early seventeenth-century Baptists were thought to be influenced by **continental Anabaptists**..

Gemini-2.0-preview's Deep Thinking
The Dashnaks' objectives were:1. Unifying various small groups in the Ottoman Empire advocating for reform.2. Defending Armenian villages from massacres.3. Creating a f̈ree, independent and unifiedÄrmenia (though they sometimes pursued autonomy instead)..

GLM-zero-preview's Deep Thinking
###Thinking Let's tackle this question about ripple current. We're asked for the name given to the AC component of an applied source having either constant or varying frequency, particularly in the context of capacitors and power supplies.The text provides a clear definition: R̈ipple current is the AC component of an applied source (often a switched-mode power supply) whose frequency may be constant or varying.Ït goes on to explain that ripple current causes heat generation within the capacitor due to dielectric losses and the internal resistance, known as the equivalent series resistance (ESR). The text also mentions that certain types of capacitors, like tantalum and aluminum electrolytic capacitors, have specified ratings for maximum ripple current.Wait, just to be absolutely sure, let's consider if there might be other terms that fit. Could it be something like ÄC ripple,c̈urrent ripple,ör just r̈ipple?̈ While these terms are related, the text specifically uses r̈ipple currentẅhen referring to the AC component of the applied source.I briefly considered whether ÄC ripplem̈ight be the answer, but the text consistently refers to it as r̈ipple current.S̈imilarly, c̈urrent ripples̈ounds a bit redundant, and r̈ippleälone is too vague. It's important to use the term as it's presented in the context provided.Just to double-check, let's revisit the key sentence: R̈ipple current is the AC component of an applied source (often a switched-mode power supply) whose frequency may be constant or varying.T̈his directly answers the question.

Deep Thinking DROP's example is as follows:

DROP dataset's Deep Thinking

QWQ-32B-preview's Deep Thinking
So I'm trying to figure out how many sons Sam Fang Kaen had before Thao Lok. Let's see what the text says.It mentions that Thao Lok was the sixth son of Sam Fang Kaen. So if he was the sixth son, that means Sam Fang Kaen had five sons before him, right? Because sixth comes after five.Let me just double-check. If you count sons, first, second, third, fourth, fifth, and then sixth, so Thao Lok

being the sixth means there are five others before him.So, the number of sons Sam Fang Kaen had before Thao Lok is five.Wait a minute, is there any other information in the text that might suggest otherwise? Let's read the relevant part again.ẗhe sixth son of the Lan Na king Sam Fang Kaen, Thao Lok, ascended the throne in the aftermath of coup against his father in 1441.Ÿes, that's clear. Thao Lok was the sixth son.

OpenAI-o1-preview's Deep Thinking
To find the percentage of people who were **not** of English ancestry in the county, we subtract the percentage of English ancestry from the total population percentage.Given:- Percentage of people with English ancestry = 11.0%Therefore:- Percentage of people **not** of English ancestry = 100% - 11.0% = **89.0%**So, **89.0%** of the population were not of English ancestry.boxed89%

Deepseek-R1's Deep Thinking
<think>Okay, let's see. The question is asking how many 1-yard touchdowns there were in the game described. I need to go through the text carefully and count each instance where a touchdown was scored from 1 yard out.Starting with the first quarter: the Panthers scored a 19-yard field goal. That's not a touchdown, so no count there.Second quarter: Panthers increased their lead with a 5-yard TD pass from Testaverde to Rosario. That's a 5-yard touchdown, so not 1 yard. Then Marshall returned an interception 67 yards for a touchdown. That's a defensive touchdown, but the distance here is 67 yards, so again, not a 1-yard TD.Third quarter: 49ers get on the board. Dilfer throws a 20-yard TD pass to Battle. That's 20 yards, so no. Then Dilfer throws a 1-yard TD pass to Walker. Okay, that's one 1-yard TD. So that's the first one.Afterwards, Panthers' DeShaun Foster gets a 1-yard TD run. That's the second 1-yard TD.Fourth quarter: Testaverde completes a 1-yard TD pass to Jeff King. That's the third 1-yard TD.So adding those up: 1 (Dilfer to Walker) + 1 (Foster run) + 1 (Testaverde to King) = 3 total 1-yard touchdowns.Wait, let me double-check to make sure I didn't miss any. Let's go through again:- 1st quarter: FG, no TD.- 2nd quarter: 5-yard TD pass and 67-yard interception return. No 1-yarders.- 3rd quarter: 20-yard TD (Battle), 1-yard TD (Walker), then Foster 1-yard run. So two in the third.- 4th quarter: 1-yard TD pass (King). That's one more.Yes, that's three total.

Deepseek-V3's Deep Thinking
To solve the problem, we need to find the prices of the three specified items and then sum them up.1. **First paperback edited by Lin Carter**: The first three paperbacks edited by Lin Carter were priced at **$2.50**.2. **Second Bellerophon issue**: The two Bellerophon issues were priced at **$2.50** and **$2.95**. The second issue is **$2.95**.3. **Fall 1988 issue of the Terminus Weird Tales**: The Fall 1988 issue of the Terminus Weird Tales was priced at **$4.00**.Now, add the prices together: [2.50 + 2.95 + 4.00 = 9.45].

Gemini-2.0-preview's Deep Thinking
The text states that Jon Kitna completed a 60-yard TD pass to WR Roy Williams. Michael Vick completed a 19-yard TD pass to TE Alge Crumpler. Kevin Jones had a 35-yard TD run and Warrick Dunn had a 1-yard TD run.

GLM-zero-preview's Deep Thinking
###ThinkingLet's tackle this problem step by step. The text mentions that the Eagles lost to the Arizona Cardinals with a score of 40-17. To find out how many points the Eagles lost by, we need to calculate the difference between the Cardinals' score and the Eagles' score.So, we subtract the Eagles' score from the Cardinals' score:40 (Cardinals) - 17 (Eagles) = 23Wait, let's make sure we're interpreting this correctly. The problem asks, Ḧow many points did the Eagles lose by?ẅhich is essentially asking for the margin of defeat. Subtracting the Eagles' score from the Cardinals' score gives us that margin.Just to be thorough, let's consider any other information in the text to

ensure we’re not missing anything. The Eagles wore their black uniforms, the Cardinals clinched a playoff spot, and rookie back David Johnson had an impressive game. There’s also mention of the Redskins winning and the Eagles being two games down in the East, but none of this additional information affects the score of this particular game between the Eagles and the Cardinals.Could we be overthinking this? The problem seems straightforward, and the other details are likely just setting the context. We could consider the possibility that the question is asking for something else, but the wording Ḧow many points did the Eagles lose by?c̈learly points to the margin of defeat.Let’s double-check the subtraction to be absolutely sure:40 - 17 = 23Yes, the Eagles lost by 23 points.Even though there’s a lot of extra information about the game, we don’t need any of it to answer the question. The only numbers that matter are the final scores.

F Details of Results for Each Reasoning LLM

Table 6: Details of Results of Mixtral-8x7B.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	83	31	77	27	25	75	88	72
Deepseek-R1	87 (+4)	57 (+26)	91 (+14)	62 (+35)	27 (+2)	89 (+14)	88 (+0)	90 (+18)
Deepseek-V3	86 (+3)	56 (+25)	88 (+11)	74 (+47)	44 (+19)	90 (+15)	91 (+3)	88 (+16)
QWQ-32B-Preview	87 (+4)	57 (+26)	95 (+18)	63 (+36)	48 (+23)	90 (+15)	87 (-1)	85 (+13)
Gemini-2.0-Flash	88 (+5)	56 (+25)	95 (+18)	82 (+55)	31 (+6)	86 (+11)	94 (+6)	87 (+15)
GLM-Zero-Preview	91 (+8)	59 (+28)	96 (+19)	53 (+26)	44 (+19)	88 (+13)	82 (-6)	86 (+14)
o1-preview	87 (+4)	54 (+23)	90 (+13)	89 (+62)	32 (+7)	88 (+13)	87 (-1)	87 (+15)
o3 - mini	91 (+8)	82 (+51)	74 (-3)	87 (+60)	37 (+12)	83 (+8)	88 (+0)	85 (+13)

Table 7: Details of Results of LLaMA3.1-8B.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	86	56	95	47	66	86	92	86
Deepseek-R1	89 (+3)	59 (+3)	93 (-2)	62 (+15)	83 (+17)	89 (+3)	91 (-1)	94 (+8)
Deepseek-V3	86 (+0)	62 (+6)	95 (+0)	59 (+12)	71 (+5)	89 (+3)	94 (+2)	94 (+8)
QWQ-32B-Preview	88 (+2)	64 (+8)	92 (-3)	65 (+18)	77 (+11)	89 (+3)	88 (-4)	92 (+6)
Gemini-2.0-Flash	90 (+4)	64 (+8)	95 (+0)	77 (+30)	83 (+17)	85 (-1)	94 (+2)	91 (+5)
GLM-Zero-Preview	91 (+5)	61 (+5)	83 (-12)	29 (-18)	77 (+11)	85 (-1)	93 (+1)	92 (+6)
o1-preview	91 (+5)	57 (+1)	92 (-3)	67 (+20)	62 (-4)	89 (+3)	88 (-4)	92 (+6)
o3-mini	82 (-4)	57 (+1)	82 (-13)	73 (+26)	54 (-12)	85 (-1)	90 (-2)	91 (+5)

Table 8: Details of Results of LLaMA3.1-70B.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	92	62	98	68	35	94	91	82
Deepseek-R1	88 (-4)	62 (+0)	96 (-2)	80 (+12)	85 (+50)	91 (-3)	90 (-1)	88 (+6)
Deepseek-V3	86 (-6)	65 (+3)	95 (-3)	84 (+16)	80 (+45)	90 (-4)	92 (+1)	90 (+8)
QWQ-32B-Preview	90 (-2)	63 (+1)	97 (-1)	83 (+15)	76 (+41)	91 (-3)	89 (-2)	86 (+4)
Gemini-2.0-Flash	90 (-2)	67 (+5)	95 (-3)	86 (+18)	80 (+45)	88 (-6)	94 (+3)	86 (+4)
GLM-Zero-Preview	92 (+0)	65 (+3)	95 (-3)	80 (+12)	74 (+39)	88 (-6)	92 (+1)	88 (+6)
o1-preview	92 (+0)	60 (-2)	92 (-6)	90 (+22)	60 (+25)	84 (-10)	90 (-1)	89 (+7)
o3-mini	91 (-1)	73 (+11)	83 (-15)	89 (+21)	66 (+31)	82 (-12)	91 (+0)	92 (+10)

Table 9: Details of Results of Qwen2.5-7B.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	86	53	95	44	79	92	92	82
Deepseek-R1	88 (+2)	60 (+7)	96 (+1)	56 (+11)	82 (+3)	89 (-3)	89 (-3)	91 (+9)
Deepseek-V3	85 (-1)	66 (+13)	95 (+0)	74 (+30)	89 (+10)	90 (-2)	92 (+0)	90 (+8)
QWQ-32B-Preview	88 (+2)	64 (+11)	96 (+1)	50 (+6)	81 (+2)	90 (-2)	91 (-1)	87 (+5)
Gemini-2.0-Flash	91 (+5)	64 (+11)	94 (-1)	70 (+26)	82 (+3)	88 (-4)	92 (+0)	92 (+10)
GLM-Zero-Preview	92 (+6)	64 (+11)	97 (+2)	61 (+17)	80 (+1)	86 (-6)	87 (-5)	89 (+7)
o1-preview	92 (+6)	62 (+9)	92 (-3)	86 (+42)	81 (+2)	85(-7.00)	89 (-3)	88 (+6)
o3-mini	82 (-4)	55 (+2)	91 (-4)	86 (+42)	73 (-6)	86 (-6)	87 (-5)	88 (+6)

Table 10: Details of Results of Gemma-2-9b-It.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	84	52	90	44	62	82	84	84
Deepseek-R1	89 (+5)	59 (+7)	89 (-1)	60 (+16)	80 (+18)	88 (+6)	91 (+7)	90 (+6)
Deepseek-V3	87 (+3)	63 (+11)	93 (+3)	79 (+35)	70 (+8)	90 (+8)	91 (+7)	88 (+4)
QWQ-32B-Preview	88 (+4)	62 (+10)	99 (+9)	72 (+28)	78 (+16)	90 (+8)	84 (+10)	87 (+3)
Gemini-2.0-Flash	90 (+6)	66 (+14)	95 (+5)	83 (+39)	69 (+7)	87 (+5)	93 (+9)	85 (+1)
GLM-Zero-Preview	88 (+4)	58 (+6)	94 (+4)	60 (+16)	78 (+16)	85 (+3)	87 (+3)	86 (+2)
o1-preview	92 (+8)	61 (+9)	93 (+3)	88 (+44)	60 (-2)	86 (+4)	87 (+3)	86 (+2)
o3-mini	82 (-2)	55 (+3)	91 (+1)	88 (+44)	63 (+1)	84 (+2)	85 (+1)	86 (+2)

Table 11: Details of Results of Gemini1.5-Flash.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	86	42	93	77	79	85	95	87
Deepseek-R1	88 (+2)	60 (+18)	97 (+4)	78 (+1)	90 (+11)	90 (+5)	93 (-2)	91 (+4)
Deepseek-V3	88 (+2)	62 (+20)	94 (+1)	79 (+2)	77 (-2)	87 (+2)	92 (-3)	90 (+3)
QWQ-32B-Preview	88 (+2)	60 (+18)	97 (+4)	82 (+5)	88 (+9)	90 (+5)	90 (-5)	90 (+3)
Gemini-2.0-Flash	91 (+5)	61 (+19)	94 (+1)	90 (+13)	78 (-1)	86 (+1)	91 (-4)	91 (+4)
GLM-Zero-Preview	91 (+5)	61 (+19)	97 (+4)	78 (+1)	84 (+5)	86 (+1)	87 (-8)	89 (+2)
o1-preview	89 (+3)	59 (+17)	91 (-2)	87 (+10)	83 (+4)	82 (-3)	91 (-4)	90 (+3)
o3-mini	73 (-13)	55 (+13)	91 (-2)	87 (+10)	79 (+0)	83 (-2)	89 (-6)	92 (+5)

Table 12: Details of Results of Gemini1.5-Pro.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	91	57	96	80	86	88	90	87
Deepseek-R1	87 (-4)	61 (+4)	97 (+1)	83 (+3)	93 (+7)	90 (+2)	91 (+1)	91 (+4)
Deepseek-V3	88 (-3)	65 (+8)	96 (+0)	82 (+2)	93 (+7)	90 (+2)	92 (+2)	92 (+5)
QWQ-32B-Preview	90 (-1)	67 (+12)	98 (+2)	85 (+5)	91 (+5)	90 (+2)	92 (+2)	88 (+1)
Gemini-2.0-Flash	91 (+0)	60 (+3)	96 (+0)	89 (+9)	89 (+3)	86 (-2)	94 (+4)	89 (+2)
GLM-Zero-Preview	92 (+1)	62 (+5)	98 (+2)	83 (+3)	86 (+0)	88 (+0)	93 (+3)	89 (+2)
o1-preview	91 (+0)	65 (+8)	91 (-5)	87 (+7)	88 (+2)	88 (+0)	92 (+1)	92 (+5)
o3-mini	82 (-9)	73 (+16)	100 (+4)	90 (+10)	84 (-2)	87 (-1)	93 (+3)	93 (+6)

Table 13: Details of Results of Claude-3.5-Sonnet.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	92	53	99	72	84	93	93	86
Deepseek-R1	82 (-10)	62 (+9)	91 (-8)	76 (+4)	70 (-14)	90 (-3)	93 (+0)	93 (+7)
Deepseek-V3	83 (-9)	62 (+9)	89 (-10)	79 (+7)	74 (-10)	89 (-4)	95 (+2)	97 (+11)
QWQ-32B-Preview	91 (-1)	67 (+14)	97 (-10)	76 (+4)	66 (-18)	89 (-4)	92 (-1)	93 (+7)
Gemini-2.0-Flash	90 (-2)	69 (+16)	94 (-5)	87 (+15)	75 (-9)	89 (-4)	96 (+3)	93 (+7)
GLM-Zero-Preview	91 (-1)	63 (+10)	93 (-6)	79 (+7)	76 (-8)	85 (-8)	95 (+2)	92 (+6)
o1-preview	88 (-4)	64 (+11)	92 (-7)	75 (+3)	88 (+4)	89 (-4)	95 (+2)	95 (+9)
o3-mini	91 (-1)	82 (+29)	91 (-8)	85 (+13)	82 (-2)	89 (-4)	92 (-1)	95 (+9)

Table 14: Details of Results of GPT-3.5-Turbo.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	86	60	88	45	72	81	91	70
Deepseek-R1	89 (+3)	59 (-1)	92 (+4)	66 (+21)	59 (-13)	91 (+10)	93 (+2)	90 (+20)
Deepseek-V3	86 (+0)	63 (+3)	95 (+7)	73 (+28)	66 (-6)	90 (+9)	94 (+3)	90 (+20)
QWQ-32B-Preview	89 (+3)	65 (+5)	97 (+9)	65 (+20)	50 (-22)	90 (+9)	88 (-3)	86 (+16)
Gemini-2.0-Flash	91 (+5)	63 (+3)	95 (+7)	86 (+41)	56 (-16)	87 (+6)	88 (-3)	91 (+21)
GLM-Zero-Preview	92 (+6)	62 (+2)	98 (+10)	62 (+17)	56 (-16)	89 (+8)	92 (+1)	92 (+22)
o1-preview	92 (+6)	59 (-1)	92 (+4)	89 (+44)	53 (-19)	85 (+4)	90 (-1)	90 (+20)
o3-mini	91 (+5)	66 (+6)	83 (-5)	79 (+24)	65 (-7)	82 (+1)	88 (-3)	91 (+21)

Table 15: Details of Results of GPT-4o.

Type	OpenBookQA	HellaSwag	GSM8K	MATH	HumanEval	SST - 2	SQUAD	DROP
Base	92	66	94	49	81	87	93	86
Deepseek-R1	82 (-10)	62 (-4)	97 (+3)	61 (+12)	73 (-8)	90 (+3)	91 (-2)	93 (+7)
Deepseek-V3	78 (-14)	63 (+25)	95 (+1)	63 (+14)	84 (+3)	90 (+3)	97 (+4)	96 (+10)
QWQ-32B-Preview	89 (-3)	62 (-4)	98 (+4)	62 (+13)	82 (+1)	92 (+4)	90 (+3)	90 (+4)
Gemini-2.0-Flash	88 (-4)	61 (-5)	94 (+0)	72 (+23)	79 (-2)	87 (+0)	95 (+2)	94 (+8)
GLM-Zero-Preview	88 (-4)	63 (-3)	98 (+4)	59 (+10)	78 (-3)	89 (+2)	96 (+3)	96 (+10)
o1-preview	91 (-1)	63 (-3)	92 (-2)	74 (+25)	66 (-15)	87 (+0)	91 (-2)	94 (+8)
o3-mini	91 (-1)	73 (+7)	91 (-3)	84 (+35)	76 (-5)	85 (-2)	91 (-2)	94 (+8)