GSM-∞: How Do your LLMs Behave over Infinitely Increasing Reasoning Complexity and Context Length?

Yang Zhou [1] **Hongyi Liu** [1] **Zhuoming Chen** [1] **Yuandong Tian** [2] **Beidi Chen** [1]

Abstract

Recently, long-context large language models (LLMs) have shown strong performance in information retrieval and long-document QA. However, to tackle the most challenging intellectual problems, LLMs must reason effectively in long and complex contexts (e.g., frontier mathematical research). Studying how LLMs handle increasing reasoning complexity and context length is essential, yet existing benchmarks lack a solid basis for quantitative evaluation. Inspired by the abstraction of GSM-8K problems as computational graphs—and the ability to introduce noise by adding unnecessary nodes and edges—we develop a grade-school math problem generator capable of producing arithmetic problems with infinite difficulty and context length under fine-grained control. Using our newly synthesized GSM-∞ benchmark, we comprehensively evaluate existing LLMs. We find a consistent sigmoid decline in reasoning performance as complexity increases, along with a systematic inference scaling trend: exponentially increasing inference computation yields only linear performance gains. These findings underscore the fundamental limitations of current long-context LLMs and the key challenges in scaling reasoning capabilities. Our GSM-∞ benchmark provides a scalable and controllable testbed for systematically studying and advancing LLM reasoning in long and complex contexts.

1. Introduction

Recently, state-of-the-art long-context LLMs (Team et al., 2024; MiniMax et al., 2025) have achieved astonishing performance in a tremendously long context, where Team et al. (2024) achieves near-perfect performance in 10M multi-modal retrieval and long document QA. However, for long-context LLMs to contribute to cutting-edge mathematical and scientific discoveries or function as autonomous agents, they must be capable of processing dense, complex information and reason through multi-step tasks. For instance, Sir Andrew Wiles' proof (Wiles, 1995) of Fermat's Last Theorem in 1995 spans more than 88K highly compact tokens with deep logical connections, making context-level RAG (Lewis et al., 2021) **insufficient** and highlighting the need for long-context LLMs. Therefore, it is crucial to benchmark and facilitate long-context LLMs for complex reasoning and high-density information processing.

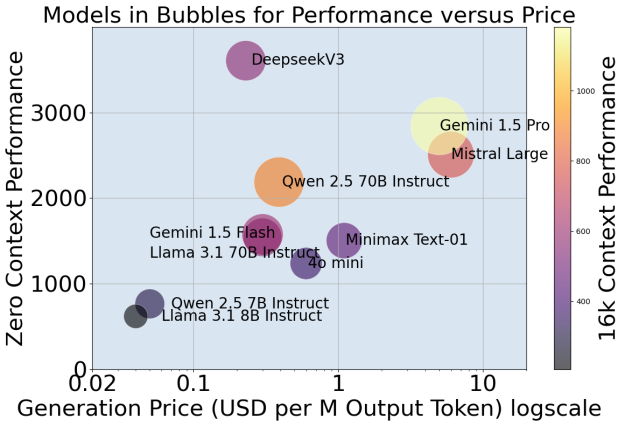

Figure 1. Evaluation of 10 powerful LLMs on GSM-∞, comparing API generation cost (horizontal axis) with zero-context reasoning ability (vertical axis). Bubble size represents reasoning performance at a 16K context length.

Although widely used, current long-context benchmarks do not fully capture the true potential of long-context LLMs (Yu et al., 2024a; Li et al., 2024a;b), making it challenging to measure their progress toward advanced intellectual agents. It is mainly due to the following three reasons: (1) **Low Complexity.** Many long-context benchmarks, such as LongBench (Bai et al., 2025; 2024) and most tasks in RULER (Hsieh et al., 2024b), focus on retrieval or summarization, which involve low reasoning complexity. Similar to (Yu et al., 2024b), we found that simple context-level RAG achieves on-par or even better results than long-context LLMs (shown in Figure 2). (2) **Detectable Noise.** Many

[1]Carnegie Mellon University [2]Meta GenAI. Correspondence to: Beidi Chen <beidic@andrew.cmu.edu>.

Proceedings of the 2nd Workshop on Long-Context Foundation Models, Vancouver, Canada. 2025. Copyright 2025 by the author(s).

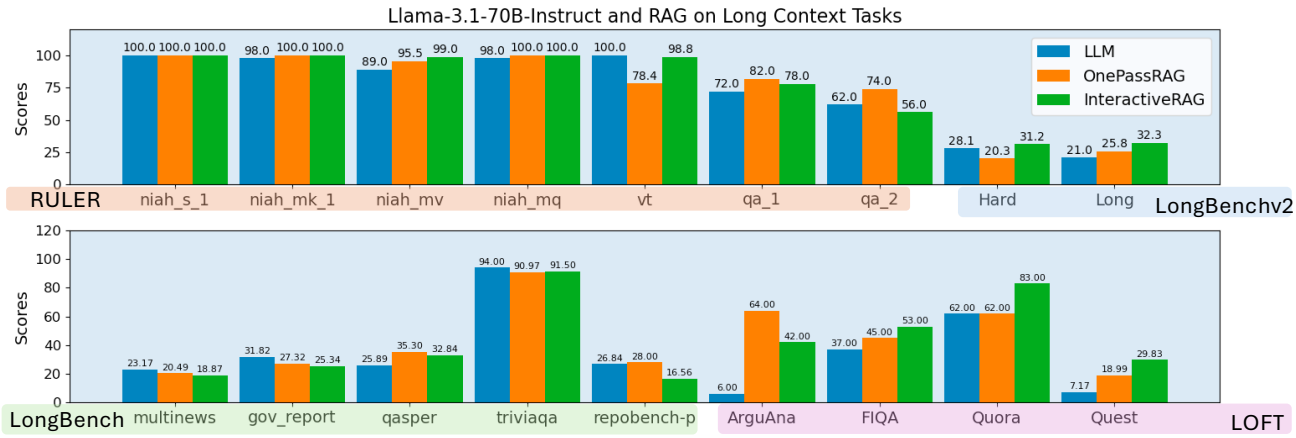

Figure 2. Study of Llama-3.1-70B-Instruct with Passive RAG (referred to as OnePassRAG) and Active RAG (referred to as InteractiveRAG) on popular long-context benchmarks: RULER (at 64K context length), LongBench (>8K), LongBenchV2, and LOFT (128K context length). RAG is under the 2048 retrieved token budget, and the decoder used for the RAG is Llama-3.1-70B-Instruct. RAGs generally have robust performance, on par with the corresponding LLMs, showing that previous long-context benchmarks are either too simple in reasoning complexity or contain detectable noise.

tasks are innately short-context but are bloated into longer context through semantically irrelevant filler text (Kuratov et al. (2024) and variable-tracing in Hsieh et al. (2024b)), which is easily distinguished by a retriever of context-level RAG. (3) **Low Resource.** While complex long-context tasks exist, such as long code completion (Loughridge et al., 2024), they lack sufficient high-quality examples. This scarcity limits test diversity and fine-grained difficulty assessment, reducing their effectiveness in model evaluation. **Figure 2 presents a motivation where we should simple-to-build and cheap RAG methods achieve on-par and sometimes even better performance on most mainstream long-context tasks compared to LC-LLMs. Due to space limitation, detailed problem formulation is presented in Appendix A.**

Ideally, a long-context reasoning benchmark should (1) offer controllable and **scalable complexity**, (2) incorporate **hard-to-distinguish noise**, and (3) support **infinite data** generation for continuous and adaptable evaluation. Inspired by Delétang et al. (2023); Ye et al. (2024a), we model reasoning problems as computation graphs enriched with language semantics. By adjusting their structure and complexity, we gain fine-grained control over reasoning difficulty and enable infinite scaling. Instead of inserting semantically irrelevant filler text, noise is introduced as additional nodes within the core graph, strategically connected to existing nodes without contributing to the necessary reasoning steps for solving the tasks. This design enables the generation of arbitrarily long test examples while making it challenging for context-level RAG to differentiate relevant information from noise in the raw text.

However, several technical challenges must be addressed to construct a practical benchmark. First, computation graphs must be effectively translated into natural language to ensure compatibility with LLMs. Second, the benchmark should disentangle LLMs' reasoning abilities from their prior knowledge. Third, ensuring diverse reasoning patterns—including variations in entities and relationships—is crucial for fair and comprehensive evaluation.

We introduce GSM-∞, a long-context benchmarking framework that scales and controls reasoning complexity and noise through fine-grained manipulation of computation graphs, enabling their translation into diverse, human- and LLM-readable problems (Short example in Figure 5(b)). **Due to space limitation, we present the detailed data construction process and generation in the Appendix B and C.** The resulting examples can scale up in both reasoning complexity and context length, which is RAG-insolvable. (detailed in Figure 3(d) and 6)

We conduct a comprehensive evaluation of 18 state-of-the-art LLMs on zero-noise problems and 10 LLMs on various noise-injected tasks using GSM-∞. In zero-noise settings, recent reasoning-optimized LLMs demonstrate substantial improvements over their non-reasoning counterparts. Notably, Deepseek-R1 (DeepSeek-AI et al., 2025) achieves an average AUC score nearly four times higher than previous SOTA models. However, in noise-injected scenarios, LLMs exhibit varying degrees of performance degradation. Our analysis reveals several key observations: (1) LLM performance Decay with reasoning complexity is in a sigmoid-like pattern; (2) LLM Performance degradation intensifies as context length increases within the same difficulty level;

Table 1. 18 selected models are evaluated on GSM-∞ zero-noise benchmarks using Area-Under-Curve (AUC), which is computed by taking the Riemann Sum of accuracy versus op count from 2 to when the model accuracy drops below 5%. We also present detailed statics of the first op number for the model to have an accuracy lower than 50%, 10%, and the average accuracy of the first 30 ops settings. Besides, we also highlight the reasoning models, linear attention hybrid models, and SSM hybrid models.

Models	Three Subtasks			Detailed Statistics on Hard Subtask			Score
	Symbolic	Medium	Hard	1st<50% op	1st<10% op	Avg. Acc op≤30	Average↑
deepseek-r1	7280.0	9750.85	8573.8	100	>130	0.9427	8534.88
o1-mini	5060.0	6054.91	3738.43	50	90	0.8397	4951.11
deepseek-v3	4310.0	4100.81	2407.86	24	55	0.6669	3606.22
qwq-32b-preview	3530.0	3205.75	1846.19	21	50	0.5403	2860.65
gemini-1.5-pro-002	2547.0	3659.59	2318.28	26	45	0.6924	2841.62
claude-3.5-sonnet	2161.0	3281.8	2115.79	26	40	0.6758	2519.53
mistral-large-2411	2332.5	2879.92	2310.49	24	50	0.6645	2507.64
qwen-2.5-72b-instruct	2048.0	2496.81	2016.38	21	40	0.5433	2187.06
gpt-4o-2024-11-20	2379.0	2457.37	1451.54	18	30	0.5064	2095.97
gemini-1.5-flash-002	1970.0	1478.75	1274.25	13	30	0.4460	1574.33
llama-3.1-70b-instruct	1769.0	1650.25	1205.25	15	30	0.4314	1541.50
minimax-text-01	1618.5	1712.64	1178.51	14	30	0.4213	1503.22
llama-3.1-405b-instruct	1557.0	1321.54	950.0	11	20	0.3409	1276.18
gpt-4o-mini	1389.0	1406.5	913.89	12	22	0.3094	1236.46
claude-3.5-haiku	897.0	1053.16	784.34	10	22	0.2910	911.50
qwen-2.5-7b-instruct	786.95	886.75	618.5	7	19	0.2257	764.07
llama-3.1-8b-instruct	462.0	786.5	606.5	6	17	0.2186	618.30
jamba-1.5-large	856.0	485.13	466.4	6	26	0.1828	602.51

(3) LLM consistently performs better on forward-thinking tasks than on backward-thinking ones; (4) Repeated sampling clearly shows that performance improves linearly with increased inference steps, but at an exponentially growing computation cost.

2. Evaluation

We evaluated 18 enormous and powerful LLMs on zero-noise problems, resulting in Table 1, while 10 models are evaluated on the long context as shown in Table 2.

From Table 1, we can see that the score separates these LLMs into clear groups. Reasoning models (R1 and o1-mini) are significantly ahead of the rest of non-reasoning LLMs. Also, similar things can be discussed when comparing 70B and 7B level models. In Table 2, we see that the models show a very different decay pattern, while Gemini-1.5-pro is significantly ahead of the rest of the other models.

Table 2. 10 selected models are evaluated on GSM-∞ Long Context benchmarks using Average AUC of Symbolic, Medium, and Hard. We evaluated models on 8K, 16K, and 32K context. Although our pipeline is capable of generating longer problems, the resource required to go further for larger models beyond our acceptance, while smaller models effectively has completely failed.

Model	8K	16K	32K	AVG↑
gemini-1.5-pro-002	1182.43	896.31	812.96	963.9
qwen-2.5-72b-instruct	927.33	681.53	563.65	724.17
mistral-large-2411	914.49	563.73	319.21	599.14
deepseek-v3	935.10	477.02	313.66	575.2
gemini-1.5-flash-002	673.88	476.72	377.38	509.3
llama-3.1-70b-instruct	479.00	394.50	355.5	409.67
minimax-text-01	481.32	359.56	325.95	388.94
gpt-4o-mini	401.00	337.81	275.63	338.15
qwen-2.5-7b-instruct	248.00	211.50	196.17	218.56
llama-3.1-8b-instruct	183.67	149.50	109.45	147.54

2.1. LLM Performance Degradation Can be Modeled Using Sigmoid Function

Our construction of GSM-∞ enables precise measurement of LLM performance across fine-grained difficulty levels. For each subtask, we observe a clear trend: LLM accuracy declines as the number of required operations increases. Surprisingly, most models exhibit a sigmoid-like performance decay, as shown in 3 for forward problems (see Section C.1 for definitions). Under the Medium subtask, LLM performance aligns remarkably well with a sigmoid function curve, with $R^2 > 0.98$. The pattern is intuitive: at low operation counts, accuracy remains near 1.0; as complexity increases, performance first decays gradually, then drops sharply toward zero, where LLMs effectively fail to solve the problems. The score eventually stabilizes near zero.

2.2. Reverse Problems are Harder to Solve for LLMs

Because the generator of GSM-∞ can generate both the "forward" and the "reverse" problems, we can compare them separately. Most LLMs perform worse in reverse problems than forward ones, shown in Figure 3(b) using

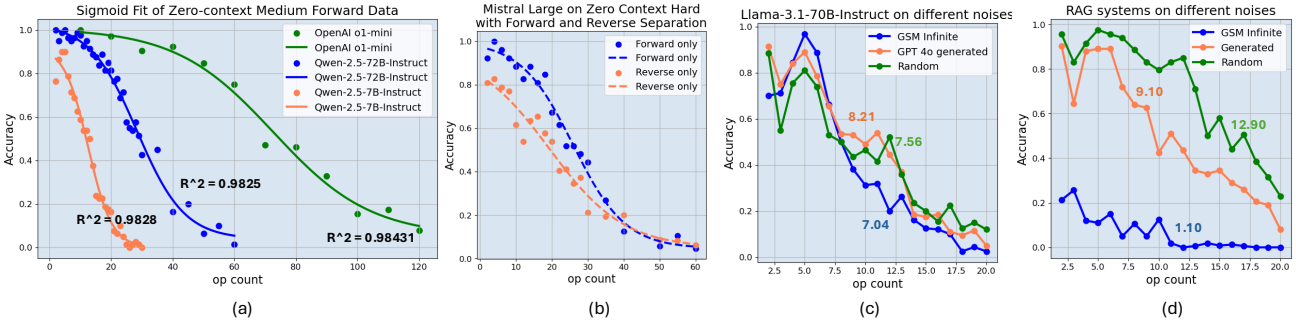

Figure 3. (a) shows three different LLMs' behavior on GSM-∞ benchmark zero-context Medium Forward, GPT o1-mini, Qwen-2.5-72B-Instruct, and Qwen-2.5-7B-Instruct. These models are drastically different in reasoning ability but have performance be modeled by sigmoid well, all with R^2 >0.98. (b) shows the gap between forward-thinking and reverse-thinking problems from Mistral Large on zero-context Hard. reverse-thinking problems are significantly harder and can be approximated by the sigmoid function that is essentially left-ward shifting from the forward sigmoid function. (c) and (d) presents RAG and corresponding LLMs' performance on different noises. Other than ours, RAG even improves performance.

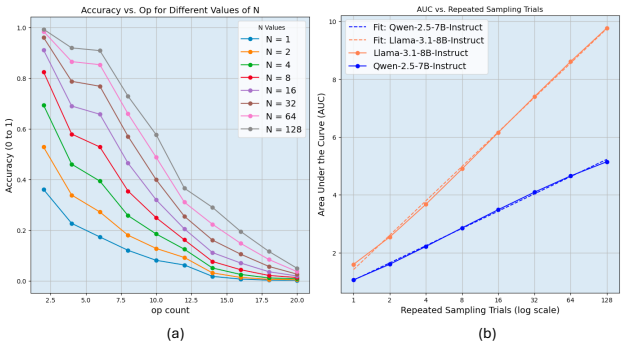

Figure 4. (a) shows repeated sampling on zero-context Hard task with Qwen-2.5-7B-Instruct; (b) shows the AUC to repeated sampling number of trials. We show that for repeated sampling, exponentially increasing inference compute only leads to a linear increase in AUC improvement.

Mistral-Large (Jiang et al., 2023a). A detailed breakdown is listed in Appendix H. Besides, LLM performance on reverse problems can also be modeled by a sigmoid mapping. Five more LLM plots are presented in Appendix H.

2.3. Long-context Degradation and Noise Ablation

We evaluate LLM performance across increasing context lengths (0, 8K, 16K, 32K) and observe a consistent decline in performance as context length increases. Notably, models exhibit different decay patterns. We present results for 10 models across 3 subtasks, each with four curves representing different context lengths. All 30 plots are in Appendix K.

We conduct an ablation study on three noise types: GSM-∞ (ours), LLM-generated, and random. For LLM-generated noise, we prompt GPT-4o to create a fake documentary-style commentary on random problems, occasionally introducing nonsensical variable mentions. For random noise, we follow Hsieh et al. (2024), using generic statements like "The sky is blue. The tree is green." We evaluate Llama-3.1-70B-Instruct and a RAG system under all three noise types in an 8K context. Interestingly, RAG outperforms long-context LLMs on LLM-generated and random noise, effectively filtering irrelevant content. However, it fails to distinguish GSM-∞ noise from essential problem statements.

2.4. Limitations of Repeated Sampling

We study both Qwen-2.5-7B-Instruct and Llama-3.1-8B-Instruct as Brown et al. (2024) best-of-N repeated sampling. Interestingly, we find that repeated sampling seems to boost the performance the most for smaller op count subsets, and the benefit of repeated sampling diminishes gradually for larger op count subsets, Qwen-2.5-7B-Instruct behavior is plotted in Figure 4(a) with different repeated trial settings.

Surprisingly, when we calculate the AUC score under every curve corresponding to each number of repeated trial settings and plot the AUC score versus the number of repeated trial settings and take the log scale of the repeated trial N, the graph is linear, as shown in Figure 4(b). (Both R-squared above 0.99) **Therefore, GSM-∞ helps reveal that Repeated Sampling leads to linear AUC Improvement from exponentially increasing inference computation cost.**

3. Conclusion

To advance their development and benchmarking, we introduce GSM-∞, a synthetic long-context reasoning benchmark generated entirely by a software-based system with fine-grained control over complexity and information density. Through extensive evaluations on GSM-∞, we uncover key insights to inform future LLM training and inference improvements.

References

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

Bai, Y., Tu, S., Zhang, J., Peng, H., Wang, X., Lv, X., Cao, S., Xu, J., Hou, L., Dong, Y., Tang, J., and Li, J. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks, 2025. URL https://arxiv.org/abs/2412.15204.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Delétang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., and Ortega, P. A. Neural networks and the chomsky hierarchy, 2023. URL https://arxiv.org/abs/2207.02098.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R.,

Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzmán, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Damlaj, I., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Prasad, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhotia, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Albiero, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Github. Needle in a haystack - pressure testing llms, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., and Ginsburg, B. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024a.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models?, 2024b. URL https://arxiv.org/abs/2404.06654.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023a.

Jiang, Z., Xu, F. F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. Active retrieval

augmented generation, 2023b. URL https://arxiv.org/abs/2305.06983.

Kamradt, G. Needle in a haystack - pressure testing llms, 2023. URL https://github.com/gkamradt/LLMTestNeedleInAHaystack/tree/main.

Kuratov, Y., Bulatov, A., Anokhin, P., Rodkin, I., Sorokin, D., Sorokin, A., and Burtsev, M. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack, 2024. URL https://arxiv.org/abs/2406.10149.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lee, J., Chen, A., Dai, Z., Dua, D., Sachan, D. S., Boratko, M., Luan, Y., Arnold, S. M. R., Perot, V., Dalmia, S., Hu, H., Lin, X., Pasupat, P., Amini, A., Cole, J. R., Riedel, S., Naim, I., Chang, M.-W., and Guu, K. Can long-context language models subsume retrieval, rag, sql, and more?, 2024. URL https://arxiv.org/abs/2406.13121.

Levy, M., Jacoby, A., and Goldberg, Y. Same task, more tokens: the impact of input length on the reasoning performance of large language models, 2024. URL https://arxiv.org/abs/2402.14848.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

Li, X., Cao, Y., Ma, Y., and Sun, A. Long context vs. rag for llms: An evaluation and revisits. *arXiv preprint arXiv:2501.01880*, 2024a.

Li, Z., Li, C., Zhang, M., Mei, Q., and Bendersky, M. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 881–893, 2024b.

Liu, H., Zaharia, M., and Abbeel, P. Ring attention with blockwise transformers for near-infinite context, 2023a. URL https://arxiv.org/abs/2310.01889.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts, 2023b. URL https://arxiv.org/abs/2307.03172.

Loughridge, C., Sun, Q., Ahrenbach, S., Cassano, F., Sun, C., Sheng, Y., Mudide, A., Misu, M. R. H., Amin, N., and Tegmark, M. Dafnybench: A benchmark for formal software verification, 2024. URL https://arxiv.org/abs/2406.08467.

MiniMax, Li, A., Gong, B., Yang, B., Shan, B., Liu, C., Zhu, C., Zhang, C., Guo, C., Chen, D., Li, D., Jiao, E., Li, G., Zhang, G., Sun, H., Dong, H., Zhu, J., Zhuang, J., Song, J., Zhu, J., Han, J., Li, J., Xie, J., Xu, J., Yan, J., Zhang, K., Xiao, K., Kang, K., Han, L., Wang, L., Yu, L., Feng, L., Zheng, L., Chai, L., Xing, L., Ju, M., Chi, M., Zhang, M., Huang, P., Niu, P., Li, P., Zhao, P., Yang, Q., Xu, Q., Wang, Q., Wang, Q., Li, Q., Leng, R., Shi, S., Yu, S., Li, S., Zhu, S., Huang, T., Liang, T., Sun, W., Sun, W., Cheng, W., Li, W., Song, X., Su, X., Han, X., Zhang, X., Hou, X., Min, X., Zou, X., Shen, X., Gong, Y., Zhu, Y., Zhou, Y., Zhong, Y., Hu, Y., Fan, Y., Yu, Y., Yang, Y., Li, Y., Huang, Y., Li, Y., Huang, Y., Xu, Y., Mao, Y., Li, Z., Li, Z., Tao, Z., Ying, Z., Cong, Z., Qin, Z., Fan, Z., Yu, Z., Jiang, Z., and Wu, Z. Minimax-01: Scaling foundation models with lightning attention, 2025. URL https://arxiv.org/abs/2501.08313.

Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., and Farajtabar, M. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

Shyam, V., Pilault, J., Shepperd, E., Anthony, Q., and Millidge, B. Tree attention: Topology-aware decoding for long-context attention on gpu clusters, 2024. URL https://arxiv.org/abs/2408.04093.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Wiles, A. Modular elliptic curves and fermat's last theorem. *Annals of mathematics*, 141(3):443–551, 1995.

Ye, T., Xu, Z., Li, Y., and Allen-Zhu, Z. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process, 2024a. URL https://arxiv.org/abs/2407.20311.

Ye, T., Xu, Z., Li, Y., and Allen-Zhu, Z. Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems, 2024b. URL https://arxiv.org/abs/2408.16293.

Yu, T., Xu, A., and Akkiraju, R. In defense of rag in the era of long-context language models. *arXiv preprint arXiv:2409.01666*, 2024a.

Yu, T., Xu, A., and Akkiraju, R. In defense of rag in the era of long-context language models, 2024b. URL https://arxiv.org/abs/2409.01666.

Zhang, X., Chen, Y., Hu, S., Xu, Z., Chen, J., Hao, M. K., Han, X., Thai, Z. L., Wang, S., Liu, Z., and Sun, M. ∞bench: Extending long context evaluation beyond 100k tokens, 2024. URL https://arxiv.org/abs/2402.13718.

A. Related Work and Problem Statement

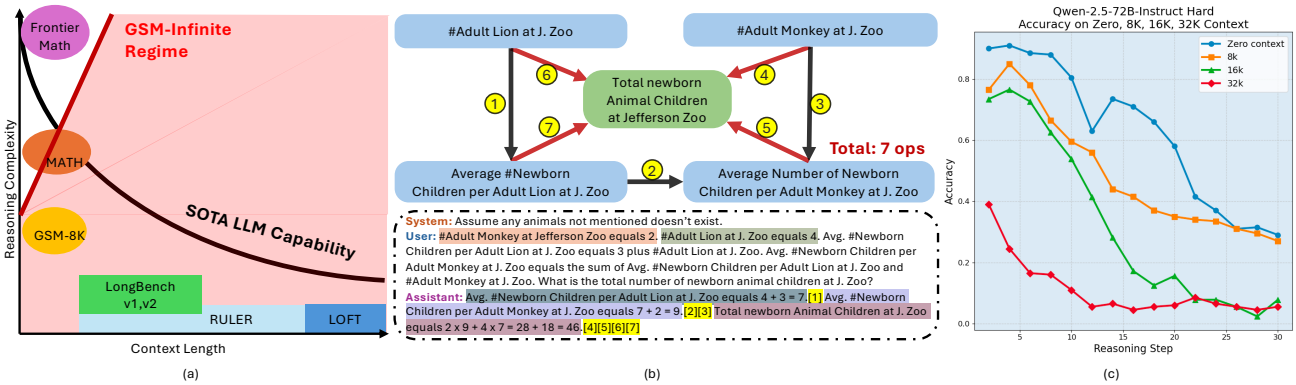

Figure 5. (a) We position existing benchmarks across the Reasoning complexity versus context length plot. Reasoning datasets are usually of very short context. Existing long context benchmarks are usually low in reasoning complexity. Our task can cover any context length that the user so chooses and can generate infinite reasoning complexity. However, for high reasoning complexity, our task needs to use a longer context for problems. Our task is shown in Red. (b) A simplified example of our dataset-building process. We first generate an interconnected computation graph, and we then based on the graph, attach real-world context to it to formulate the problem statements. (c) Shows Qwen-2.5-72B-Instruct Score decay across zero-context, 8K, 16K, and 32K.

Low Complexity. A significant portion of long-context evaluation datasets, including RULER (Hsieh et al., 2024b), LongBench (Bai et al., 2024), LongBench v2 (Bai et al., 2025), and LOFT (Lee et al., 2024), primarily assess retrieval and summarization rather than complex reasoning. Our experiments demonstrate that RAG systems achieve competitive results with Llama-3.1-70B-Instruct across these datasets. Notably, RAG outperforms LLMs in retrieval-focused tasks (e.g., RULER, LOFT) and performs comparably in text summarization and QA (most LongBench tasks), as well as structured reasoning problems such as variable tracking (RULER-vt) and code completion (LongBench-repobench-p). As shown in Figure 2, RAG methods provide a strong baseline while being substantially more efficient.

Detectable Noise. Many long-context benchmarks artificially extend short-context tasks by injecting extraneous text that does not contribute to solving the problem, allowing retrieval-based models to filter out noise effectively. In RULER's variable-tracing task with an 8192-token context, Llama-3.1-70B-Instruct achieves 100%, while OnePassRAG and InteractiveRAG reach 82.4% and 98.4%, respectively, despite using only a 2048-token retrieval budget. A detailed breakdown in Figure 6 (a) reveals that retrievers consistently identify and prioritize relevant information while disregarding injected noise. These findings indicate that existing long-context benchmarks do not adequately justify the need for expensive long-context LLMs, as RAG systems can effectively mitigate the impact of noise and achieve similar performance.

Low Resource. Many high-quality reasoning tasks heavily rely on human efforts and have test examples in limited quantity. It is infeasible to extract subsets of examples with exact op at 8 for precise LLM evaluation due to the limited number of available cases—only 26 in total, with even fewer satisfying op ≥ 8. This scarcity makes meaningful evaluation impractical. *How can we find a benchmark that contains sufficient problems at every fine-grained level of reasoning difficulty, from easy retrieval tasks to infinitely hard challenges, while providing infinitely customizable context length with high information density?*

B. Computation Graph

In this section, we explore the connection between reasoning problems and computation graphs, core ideas to scale and control reasoning complexity (Appendix B.1), and introduce challenging, indistinguishable noise (Appendix B.2) by strategically manipulating computation graph structures.

B.1. Graph Construction to Build Reasoning Problems

After running a careful study of GSM-8K problems, we draw the following crucial observations that allow us to map a randomly generated computation graph to grade-school-level math reasoning problems that cover all possible operations and relationship types.

Mapping Explicit Ops to Computation Graph - From Every operation used in the GSM-8K is one of the four "+", "-", "×", and "÷". Consider the following example when operations are presented explicitly, "Eggs cost twice as much as

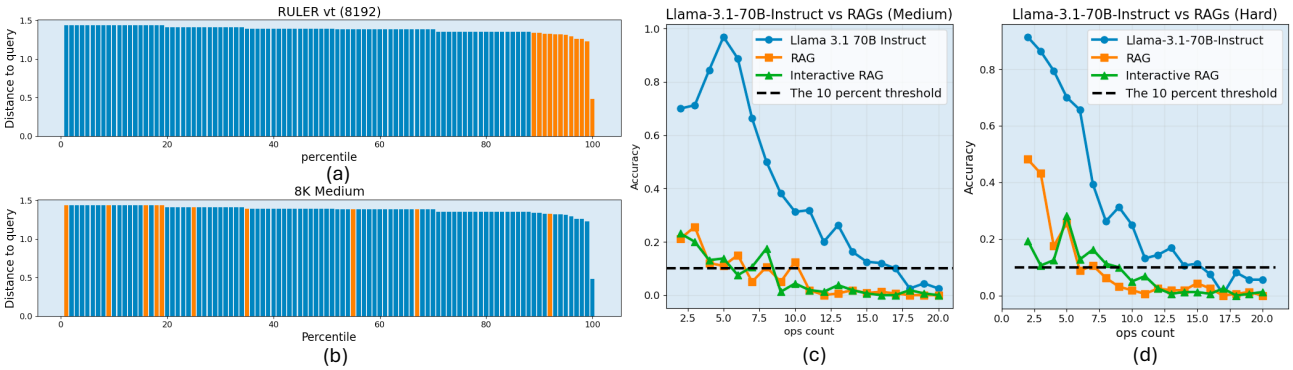

Figure 6. RAG performance on our proposed long-context benchmarks. (a) studies retriever's behavior on the first 100 chunks of a random problem in vt from RULER with 8192 context length. The chunks that need to be retrieved to solve the problem are labeled in coral, while the noise is in blue. The chunks have retriever scores ranked from large (semantically far) to small (semantically close). Retriever locates the essential chunks with high precision, classifying all necessary chunks with the right side of the spectrum; (b) contrasts vt with our long-context benchmarks, showing that the retriever cannot locate precisely which chunk to retrieve. (c) and (d) display the performance of two RAG systems on our benchmark medium and hard tasks. (Figure best viewed in color)

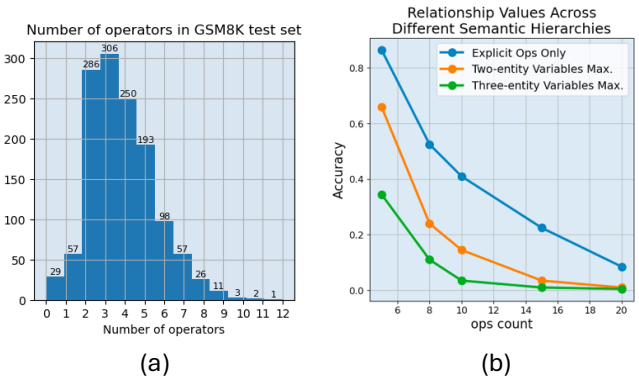

Figure 7. (a) presents a conservative estimate for each problem difficulty in GSM-8K 1.3K test set. We evaluate the difficulty of the problems by the number of operations needed to get to the final answer. The op count ranges from 2 to 12, while most are around 3-4. (b) shows the Llama-3.1-8B-Instruct performance across different semantics hierarchies, revealing the hidden reasoning difficulty innate in natural language.

tomatoes, while tomatoes cost 1 dollar each." These statements mention operations ("plus", "more", "times", etc.) can easily be abstracted out as a computation graph with variables, "dollar per egg" and "dollar per tomato", as nodes. There are two edges one pointing from "dollar per tomato" to "dollar per egg", while another one from a constant 2 to "dollar per tomato". Therefore, randomly generating a computation graph with different topology of edge connections will lead to a new reasoning problem once the natural language context are attached to the nodes of the graph.

Generating Implicit +- using Computation Graphs - On the other hand, the operations can also be presented implicitly hidden in natural language hierarchies. "Mary earns 20 dollars in the morning, while she earns 25 dollars in the afternoon. How much total she earned that day?" Although the problem doesn't explicitly mention addition, the solution has to sum up 20 and 25 to get 45. The reason is that natural language assumes a working day consists of morning and afternoon. Similarly, all four operations can be hidden in natural language hierarchies. Inspired by Ye et al. (2024a), we adopt its construct of "Abstract Parameters" and "Instance Parameters" to construct computational graphs that facilitate the generation of problem statements containing the hidden operations. Essentially, the newly added constructs can be thought of as adding the "total money" as a new node to the computational graph, which has two edges coming in, one from node "Morning money" and the other one from node "Afternoon money". But when generating the problem, we omit the description of two edges pointing to the node "total money on Friday".

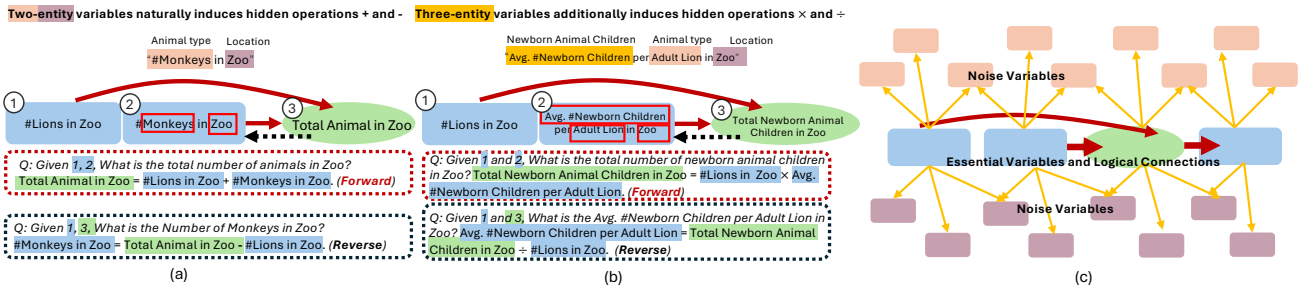

Figure 8. (a) shows a generator that focuses on only generating two-entity variables. For generating hidden operations × and ÷, we also build a generator for three-entity variables in (b). (c) We view noise generation as an extension of the essential computation graph, and we found that the spider-like topology results in a high information density long context. (Figure best viewed in color)

Generating Implicit ×÷ using Computation Graphs - The above-mentioned problem with the hidden "+" operation consists of only the "two-entity variables". "Morning Money" contains two entities, "Morning" and "Money", where in the context, "Money" is an attribute of "Morning". Same with "After- noon Money". In fact, out of all the examples we manually examined in GSM-8K, the minimum number of entities in the variable name is two. However, problems of only "two-entity" variables can only generate hidden operations of + and - but not × and ÷. For a problem to contain hidden operations ×, problems must contain variables with more than two entities in its name. For example, "Mary works 8 hours on Friday. Her hourly rate on Friday is 10 dollars. How much she will earn on Friday in total?" The variable "money per hour" contains "money", "hour", and "Friday" three entities, where "money" is an attribute of "hour", while "hour" is also an attribute of "Friday".

Our computation graph generator employs the abstract parameter construct to generate implicit operations and three-entity variables to represent multiplication operations. To maximize diversity in reasoning paths, we impose minimal restrictions on graph generation. During this process, a query node is sampled from the graph, and the corresponding topological sort list—ending with the query—ensures the shortest solution path, serving as a measure of the problem's reasoning complexity. This approach enables the generation of a vast number of synthetic graphs. Furthermore, adjusting the number of variables provides a coarse control over complexity, which is refined through precise filtering to produce well-defined subsets of graphs that meet specific operation constraints.

B.2. Noise Construction Using Computation Graph

Spider Topology - We observe that we can view noise as extending the computational graph to incorporate fake and unnecessary parameters and operators. However, two critical questions emerge. First, how to extend the computation graph without contaminating the original graph's solution and contaminations? We found that edges have to point outwards from the nodes in the original graph to the newly added noise nodes, essentially preventing the noise nodes from contributing to the core graph. Second, how to maximize the chance that RAG cannot retrieve the essential graph? It turns out interconnecting edges between newly added noise nodes won't contribute help detering RAG's retriever. We find out a simple trick works well: ensuring the majority of the added edges connect core nodes and the noise nodes contribute to a semantically close noise. We call this design Spider Topology as shown in Figure 8(c).

We evaluated the resulting noise using two RAG systems in A. The results are shown in Figures 6(c) and (d). Llama-3.1-70B-Instruct achieves drastically stronger performance than the two RAG systems on 8K 2-entity problems and 3-entity problems. We also carried out the same study before on our data set setting 2, in Figure 6(b). We found that the RAG retriever now completely cannot distinguish which essential chunks from noise chunks, showing a clear contrast with vt tasks of the same context length in (a).

C. GSM-∞

In this section, we present key techniques that enable the synthetic dataset to be diverse in operations, LLM-understandable, and enable the evaluation to be free from non-reasoning factors. Then, we present synthetic problem generators capable of generating grade-school math questions with arbitrary reasoning difficulty and context length. Thus, we generate a suite of

benchmarks called GSM-∞.

C.1. Reverse Problems

The key limitation of Ye et al. (2024a) abstract parameters and instance parameter design is that it is only able to generate problems with solutions with the "forward" and constructive ordering. Shown in the Figure 8 (a) and (b), the design dictates that the specific and detailed variables should be defined before a more abstract variable. For example, "the number of Lions in Zoo" and "the number of Monkeys in Zoo" have to be defined before "Total Animal in Zoo" is defined. The "forward" ordering leads to the inability to generate hidden '-' operations for 2-entity problems and hidden "÷" operations for 3-entity problems that require the more abstract variables, e.g. "Total Animal in Zoo", to be defined before a more specific variable, e.g. "the number of Monkeys in Zoo".

To generate all four kinds of hidden operations, we introduce a "reverse mode" to generating the computation graph. Essentially, the graph construction still continues as before: starting with specific detailed variables and growing to incorporate more abstract variables. When it completes and we know all the values of nodes in the graph, we then randomly mask out a specific initial low-level variables and force the solution to traverse in the reverse direction as in the "forward" ordering. We present the illustration of data generation in Figures 8 (a) and (b) for the 2-entity and 3-entity, respectively. However, for 3-entity problems, it can result in quadratic equations leading to multiple possible solutions. We develop some techniques that effectively reduce the probability of the situation. Details of implementation are presented in Figure 9 in Appendix.

C.2. Language Attachment through Templates

Mapping computation graphs to natural language is critical for evaluating LLMs' reasoning capabilities. To automate this process, we develop inter-swappable templates that enhance linguistic diversity while maintaining clarity. Several key considerations inform our design. First, Certain syntactic forms, such as possessive constructions (e.g., A's B), are straightforward to encode but can mislead LLMs due to their deviation from natural language. For example, South Zoo's Penguin is restructured as Penguin in South Zoo, and South Zoo's Adult Penguin's Average Number of Newborn Children becomes Average Number of Newborn Children per Adult Penguin in South Zoo. Second, to ensure minimal constraints to random graph generation, templates enforce unit consistency across two-entity and three-entity variables to enable assignment between these two. For instance, "The average number of animal children per penguin in South Zoo" must share a unit with "The number of penguins in South Zoo" to allow variable assignments. Third, to ensure real-world knowledge doesn't confuse the LLM's decision, we avoid specific real-world locations, people's names, and festival names from appearing in the template. Based on these constricts, we propose three different templates that meet real-world templates: children-animal-zoo, teachers-school-district, and awards-movies-festival. We present in Appendix I an ablation study showing that three templates are consistent in overall performance with only minor fluctuations when evaluated using Llama-3.1-8B-Instruct. At each op, equal problems are tested for both constructive ordering (forward) and reverse ordering (reverse).

C.3. Benchmark Details

With the synthetic problem generators detailed in Section C, we then use them to generate problems to build a suite of reasoning tasks with increasing complexity. For the brevity of reference, we refer to the generated problems with only explicit operations as "Easy", the generated problems with 2-entity variables at maximum as "Medium", and the generated problems with 3-entity variables at maximum as "Hard.

Ideally, when evaluating an LLM, we want to evaluate all difficulty levels, from the most basic logic complexity to when it completely fails to solve any problem. For the Easy subset of problems, it usually leads to large operation counts for powerful LLMs. However, although complexity-wise not challenging, LLMs trained with internal COT tend to generate very long arguments, saturating their API output generation limit (4K for many models). Thus, we observe a sudden decay in accuracy in large ops, not because of LLMs' ability bottlenecks, but because of the above-mentioned nuance. Thus, we make a tweak to its problem: Instead of asking the LLM to find the value of one variable, we ask the LLM to find all the variables that have some value specified, effectively increasing the difficulty of the problem.

For the modified Easy subset, we keep the generated problem in the most basic form: symbolic assignment. The typical problem statement then becomes "v1235 equals v1468 plus 1." Since the modified problem is not easier compared to Medium

and Hard, we now call it "Symbolic". For Medium and Hard, we use all three templates and mixed the generated problems together to ensure diversity. For reporting LLMs performance, we use Area Under Curve (AUC), which is computing a Riemann sum over the LLM's performance in accuracy versus number of operations from 2 to when its performance is lower than 5%.

We prepare zero-noise, 8K, 16K, and 32K in the benchmarks. The existing generation pipeline is capable of generating in > 16M context, but the smaller 70B level models effectively failed in the 32K context already, while evaluating larger ones brings cost beyond our acceptance.

D. Related Work

D.1. Long-context Language Models

Various works related to the Long-context Language Model have been proposed. Flash attention(Dao et al., 2022), Flash attention2(Dao, 2024), Ring attention(Liu et al., 2023a), and Tree attention(Shyam et al., 2024) significantly reduced the memory footprint and communication overhead for processing long context in engineering level across multiple nodes. Architectural level innovations such as sparse attentions represented by sliding window attention(Beltagy et al., 2020), are also widely used to reduce the overhead caused by the increasing sequence length. New training strategies, such as gradually extending the training context length in the final stages of pretraining have been applied to support a long context window(Dubey et al., 2024).

D.2. Long context benchmarks and tasks

There have been a quite a few works benchmarking long-context language models. Existing comprehensive benchmarks like ∞bench(Zhang et al., 2024) cover realistic tasks including document QA, summary, and synthetic tasks including information retrieval, expression calculation, extending the context length in the benchmark to over 200k tokens. ∞bench(Zhang et al., 2024) does have mathematical reasoning tasks, however the most relevant math.calc part seems to be too difficult for SOTA models to work out. Synthetic tasks often offer more control and are less affected by parametric knowledge in comparison with realistic tasks. One comprehensive synthetic benchmark is RULER(Hsieh et al., 2024a), a synthetic benchmark with tasks including retrieval, variable tracking and so on, offering some controls over context length and task complexity. Experiments with various complexities were done, but it does not provide a quantitative analysis of complexity and context length on the correctness of the task, let alone isolate two separate patterns of performance decay. Other benchmarks usually focus on simple retrieval(Github, 2023; Liu et al., 2023b), fact reasoning(Kuratov et al., 2024), the impact of long context on natural language reasoning(Levy et al., 2024) and other real-world knowledge involved tasks.

D.3. Limitation of Existing Reasoning Tasks

Popular reasoning benchmarks are loose collections of human-made problems that naturally suffer from the following limitations. Firstly, the difficulty of problems within the same benchmark varies widely. We analyzed all 1.3K test problems in the GSM8K dataset, we plot the histogram in the number of operations in Figure 7(a). The problem varied from 2 to over 12 following a skewed bell shape curve. This lack of fine-grained control makes it challenging to systematically evaluate models across incremental difficulty levels. Also, notice that the total number of problems is less than 10 for op ≥ 9, too little for stable evaluation. The lack of problem quantity on human-curated datasets eliminates the possibility of filtering out problems of each fine-grained difficulty level. Secondly, there is a significant difficulty gap between the benchmarks: GSM-8K focuses on middle school problems, MATH (Hendrycks et al., 2021) and AIME targets prospective university students, and Frontier Math challenges top-tier math graduate students. It is difficult to quantitatively determine the difference in problem difficulty between GSM-8K problems with MATH problems since MATH uses operations such as taking power or roots that are absent in GSM-8K. Similarly, it is not possible to determine the difference in complexity from MATH to Frontier Math. It is difficult to quantitatively model LLMs' performance degradation with the continuously increasing difficulty of the problem. Third, most of the existing problems have very short input prompts. On average, GSM-8K test set problems have a length of 59.96 tokens, while MATH test set problems have 67.37 tokens when using Llama 3.1 tokenizer. We have seen from A that the addition of irrelevant noise cannot meaningfully evaluate the ability to reason in a long context of LLMs.

D.4. Synthesized Datasets for long-context

Synthesized tasks are simple to build and absolutely deterministic, data contamination safe, but highly effective to evaluate certain aspects of LLM performance. Its use in long-context benchmarks is profound. Needle-in-the-haystack (Kamradt, 2023), a pioneering long-context synthesized task, now becomes the go-to task for evaluating LLM long-context retrieval ability. On the other hand, LLM reasoning benchmarks also see recent efforts in synthesized tasks. (Mirzadeh et al., 2024) recently proposes to use build synthesized dataset upon GSM8K (Cobbe et al., 2021) to study the robustness of LLM reasoning. **Part of our work draws a strong inspiration from a series of works ((Ye et al., 2024a), (Ye et al., 2024b)) which systematically studies the intricacies of decoder transformers in solving grade-school level problems.** Following their footsteps, we carefully redesign the process of generating the problems so current LLMs can solve without training, and together with thoughtful steps in noise addition, we effectively construct effective reasoning benchmarks for the long-context community.

E. Detailed Experiment Setup

E.1. RAG Experiment Setup

The RAG system contains two components, the retriever and the decoder. For the retriever, we use all-mpnet-v2-base. For the decoder, we use Llama-3.1-70B-Instruct. The context retrieval budget for all problems is 2048. We employ two different RAG methods: passive and active RAGs. Passive RAG calls the retriever once before the generation of the decoder. The retriever computes the semantic similarity or distance between each chunk of context and the query sentence. These chunks in context are then ranked from closest (most semantic similar) to furthest (least semantic similar), and depending on the retrieved context, top-k chunks are retrieved. For our study, we used the L2 distance between context chunk embeddings and query embeddings. The decoder then takes the retrieved chunks as input and then outputs its response to the query.

We use two types of RAG systems: Passive RAG that only calls the retriever once at the beginning to retrieve relevant context or Interactive RAG (Jiang et al., 2023b) in which the decoder decides when to retrieve, how many retrievals are needed, and generate a query for each retrieval. For the latter one, we restrict the decoder generation with only the latest retrieval content and its past generation.

On the other hand, active RAG shows strong performance (Jiang et al., 2023b) especially for common sense reasoning tasks. In addition to the steps in passive RAGs, the decoder is allowed to initiate additional calls to the retriever to retrieve more context by generating new queries. We follow the state-of-the-art active RAG method FLARE (Jiang et al., 2023b), which allows for 10 rounds of query, but restricts the LLM to only see its current round of retrieved context and its past rounds of generation to generate its full response.

E.2. Repeated Sampling Experiment Setup

E.2.1. Procedure

1. **Oversampling Phase**
   - Generate 256 samples per task with temperature $T = 1.0$
   - Use fixed random seeds for reproducibility
2. **Accuracy Calculation**
   - Compute per-task empirical accuracy:

$$p_{\text{task}} = \frac{\text{\# Correct Samples}}{256} \tag{1}$$

   - Estimate accuracy for N samples:

$$\text{Acc}_{\text{task}} = 1 - (1 - p_{\text{task}})^N \tag{2}$$

3. **Aggregation**
   - Average results across 80 tasks:

$$\text{Final Accuracy} = \frac{1}{80} \sum_{i=1}^{80} \text{Acc}_{\text{task}_i} \tag{3}$$

E.2.2. RATIONALE

- **Oversampling**: 256 samples reduces variance in estimating p_{task} compared to using 128 samples directly
- **Probability Formula**: Models cumulative success probability:

$$P(\geq 1 \text{ correct in } k \text{ trials}) = 1 - (1 - p)^k$$

- **Task Count**: 80 tasks per op provide stable statistics while remaining computationally feasible

F. Full Result of RAG experiments

Here, we present the full result of the RAG experiment for further analysis.

F.1. RULER

Models	s1	s2	s3	mk1	mk2	mk3	mv	mq	Context Length
Llama 3.1 70B Instruct	100	100	100	100	100	100	100	100	8k
OnePass RAG	100	100	100	100	100	96	98.5	100	8k
Interactive RAG	100	100	100	100	100	98	99	99	8k
Llama 3.1 70B Instruct	100	100	100	100	98	100	98	100	32k
OnePass RAG	100	100	100	98	100	72	99.5	97.5	32k
Interactive RAG	100	100	100	98	96	96	98.5	98.5	32k
Llama 3.1 70B Instruct	100	100	100	98	96	100	89	98	64k
OnePass RAG	100	100	100	100	100	56	95.5	100	64k
Interactive RAG	100	100	100	100	96	98	99	100	64k

Table 3: RAG vs Model (RULER NIAH)

Models	vt	cwe	fwe	qa1	qa2	Context Length
Llama 3.1 70B Instruct	100	100	96.67	84	74	8k
OnePass RAG	82.4	14.8	97.33	86	86	8k
Interactive RAG	98.4	31.2	79.33	80	68	8k
Llama 3.1 70B Instruct	100	95.2	97.33	80	66	32k
OnePass RAG	86	5.2	92	84	74	32k
Interactive RAG	98	7.6	80	78	64	32k
Llama 3.1 70B Instruct	100	6.2	95.33	72	62	64k
OnePass RAG	78.4	1.2	88.67	82	74	64k
Interactive RAG	98.8	2.6	72	78	56	64k

Table 4: RAG vs Model (RULER other subsets)

Abbreviations: s1-3 = niah_single_1-3, mk1-3 = niah_multikey_1-3, mv = niah_multivalue, mq = niah_multiquery

F.2. LongBench V2

Tasks	Overall	Easy	Hard	Short	Long
Llama 3.1 70B Instruct	30	33.3	28.1	44.7	21
OnePassRAG (budget 2048)	25	33.3	20.3	23.7	25.8
InteractiveRAG (budget 2048)	33	36.1	31.2	34.2	32.3

Table 5: RAG vs Model (LongBench V2)

F.3. LongBench

Tasks	passage_count	hotpot-qa	samsum
Llama 3.1 70B Instruct	36.0,36.0,32.0	58.87,71.22,76.44	28.88,35.95,41.48
OnePassRAG (budget 2048)	0.0,0.0,0.0	65.04,61.59,63.05	31.63,23.28,26.84
InteractiveRAG (budget 2048)	27.0,14.0,6.0	61.86,51.0,55.94	24.83,20.21,23.81

Table 6: RAG vs Model (LongBench) - Part 1

Tasks	multi-news	multifieldqa_en	gov_report
Llama 3.1 70B Instruct	27.71,24.81,23.17	57.31,51.83,64.98	34.94,34.97,31.82
OnePassRAG (budget 2048)	26.85,22.72,20.49	51.69,48.66,55.85	32.6,30.67,27.32
InteractiveRAG (budget 2048)	24.64,19.99,18.87	47.71,42.98,58.45	29.91,27.02,25.34

Table 7: RAG vs Model (LongBench) - Part 2

Tasks	qasper	passage_retrieval_en	2wikimqa
Llama 3.1 70B Instruct	50.3,46.5,25.89	100.0,100.0,100.0	74.93,64.37,59.6
OnePassRAG (budget 2048)	45.73,43.5,35.3	72.0,72.0,79.0	67.97,59.64,48.4
InteractiveRAG (budget 2048)	43.8,37.9,32.84	91.0,90.0,85.33	46.04,43.46,38.8

Table 8: RAG vs Model (LongBench) - Part 3

Tasks	triviaqa	trec	lcc	repobench-p
Llama 3.1 70B Instruct	82.0,93.6,94.0	48.0,12.0,12.0	50.14,55.0,50.04	29.98,27.82,26.84
OnePassRAG (budget 2048)	92.13,89.46,90.97	47.0,56.0,53.0	19.92,14.5,18.36	34.76,33.62,28.0
InteractiveRAG (budget 2048)	88.11,92.32,91.5	56.0,57.0,52.0	24.26,23.26,22.57	14.97,17.15,16.56

Table 9: RAG vs Model (LongBench) - Part 4

Abbreviations: The 3 data separated by commas are subsets of 0-4k, 4-8k,8k+ respectively

F.4. LOFT

Tasks	ArguAna	FEVER	FIQA	MS MARCO	NQ	Quora	SciFact
Llama 3.1 70B Instruct	0.06	0.78	0.37	0.67	0.84	0.62	0.59
OnePassRAG (budget 2048)	0.64	0.88	0.45	0.77	0.86	0.62	0.64
InteractiveRAG (budget 2048)	0.42	0.73	0.53	0.69	0.76	0.83	0.87

Table 10: RAG vs Model (LOFT) - Part 1

Tasks	Touché-2020	HotPotQA	MuSiQue	QAMPARI	QUEST
Llama 3.1 70B Instruct	0.4411	0.37	0.2	0.024	0.07166
OnePassRAG (budget 2048)	0.2529	0.455	0.2383	0.1559	0.1899
InteractiveRAG (budget 2048)	0.79	0.29	0.13	0.1539	0.2983

Table 11: RAG vs Model (LOFT) - Part 2

Abbreviations: We completed this experiment only on 128k context length because LOFT didn't release their official prompt for 32k and 1M.

G. Illustrative Problems

This section presents one representative problem from each subset (Symbolic, Medium, and Hard) defined in the appendix. These examples illustrate the variations within the benchmark. see Table 12

Table 12. Illustrative Problems from Each Subset

Feature	Symbolic	Medium	Hard
Problem	• **Symbolic (op=5):** <context>\nassign V705804 = V437110 + 1. assign V986916 = V705804. assign V873548 = 6. assign V684196 = V873548. assign V437110 = V873548.\n </context> \n\nThe context contains relationships between variables. These relationships are independent mathematical equations that are all satisfied simultaneously.\n Using only these relationships, determine which variables (if any) from which values can be derived are equal to 7.\nShow your step-by-step reasoning and calculations, and then conclude your final answer in a sentence. **Answer**: V705804,V986916.	• **Medium (op=5):** Problem: The number of adult owl in Bundle Ranch equals 2 times the number of adult eagle in Bundle Ranch. The number of adult eagle in Hamilton Farm equals the difference between the total number of adult animals in Bundle Ranch and the number of adult eagle in Bundle Ranch. The number of adult owl in Hamilton Farm equals 4 times the number of adult owl in Bundle Ranch. The number of adult eagle in Bundle Ranch equals 3. Question: What is the total number of adult animals in Bundle Ranch? **Answer**: 9.	• **Hard (op=5):** The average number of newborn children per adult blue jay in Bundle Ranch equals 2. The number of adult parrot in Bundle Ranch equals 2. The number of adult blue jay in Bundle Ranch equals 2 times the average number of newborn children per adult blue jay in Bundle Ranch. The number of adult eagle in Bundle Ranch equals 2 times the average number of newborn children per adult blue jay in Bundle Ranch. The number of adult parrot in South Zoo equals 4 times the sum of the average number of newborn children per adult eagle in Hamilton Farm, the number of adult eagle in Hamilton Farm, and the average number of newborn children per adult eagle in Hamilton Farm. The average number of newborn children per adult eagle in Hamilton Farm equals the number of adult eagle in Bundle Ranch. The number of adult eagle in Hamilton Farm equals 3. The average number of newborn children per adult parrot in Bundle Ranch equals the total number of adult animals in Hamilton Farm. The number of adult eagle in South Zoo equals 1. The average number of newborn children per adult parrot in South Zoo equals the average number of newborn children per adult parrot in Bundle Ranch. The average number of newborn children per adult eagle in Bundle Ranch equals 3 plus the average number of newborn children per adult parrot in Bundle Ranch. The average number of newborn children per adult eagle in South Zoo equals the sum of the number of adult blue jay in Bundle Ranch, the average number of newborn children per adult blue jay in Bundle Ranch, the average number of newborn children per adult parrot in Bundle Ranch, and the number of adult parrot in Bundle Ranch. Question: What is the average number of newborn children per adult eagle in Bundle Ranch? **Answer**: 6.

H. Forward and Reverse Problems Breakdown

Models	Forward Problem	Reverse Problem	Forward AUC - Reverse AUC
Llama 3.1 70B Instruct	2100.625000	1283.750000	816.875000
GPT 4O-mini	1529.725400	1267.579000	262.146400
Jamba-1.5-Large	390.380000	624.980000	-234.600000
GPT 4O	3073.997375	1952.816875	1121.180500
Mistral Large	3468.234100	2431.732450	1036.501650
Llama 3.1 8B Instruct	1030.000000	563.125000	466.875000
Claude Sonnet	3653.830050	3158.657850	495.172200
Qwen 2.5 72B Instruct	2889.375000	2141.250000	748.125000
Qwen 2.5 7B Instruct	995.625000	833.125000	162.500000
o1-mini	6517.510550	5592.307100	925.203450
Gemini 1.5 Flash	1889.375000	1153.750000	735.625000
Claude Haiku	1234.620000	873.100000	361.520000
Llama 3.1 405B Instruct	1781.400000	981.250000	800.150000
DeepseekV3	4613.125000	3713.125000	900.000000
Gemini 1.5 Pro	4204.564075	3160.574950	1043.989125
Deepseek R1	9764.950000	9750.950000	14.000000
Minimax Text-01	2148.071300	1539.415650	608.655650
Qwen-QwQ-32B-Preview	3530.000000	2846.250000	683.750000

Table 13: Medium Difference in AUC in Forward Problems and Reverse Problems

Models	Forward Problem	Reverse Problem	Forward AUC - Reverse AUC
Claude Haiku	819.240000	776.900000	42.340000
Llama 3.1 70B Instruct	1314.375000	1098.750000	215.625000
Gemini 1.5 Flash	1341.250000	1219.375000	121.875000
Minimax Text-01	1360.555000	1034.625000	325.930000
Deepseek R1	8444.500000	8756.950000	-312.450000
OpenAI o1mini	3831.381000	3645.474200	185.906800
Gemini 1.5 Pro	2255.732025	2444.270375	-188.538350
DeepseekV3	2725.085000	2109.560000	615.525000
Qwen-2.5-7B-Instruct	625.625000	630.625000	-5.000000
4O	1592.280000	1311.560000	280.720000
Llama 3.1 8B Instruct	759.375000	460.625000	298.750000
Qwen 2.5 72B Instruct	2196.875000	1895.000000	301.875000
Claude Sonnet	2242.309950	1999.998100	242.311850
4O-mini	858.400000	873.310000	-14.910000
Qwen-QwQ-32B-Preview	1878.750000	1855.625000	23.125000
Mistral Large	2570.940500	2018.469000	552.471500
Llama 3.1 405B Instruct	1215.000000	743.750000	471.250000
Jamba-1.5-Large	274.980000	699.990000	-425.010000

Table 14: Hard Difference in AUC in Forward Problems and Reverse Problems

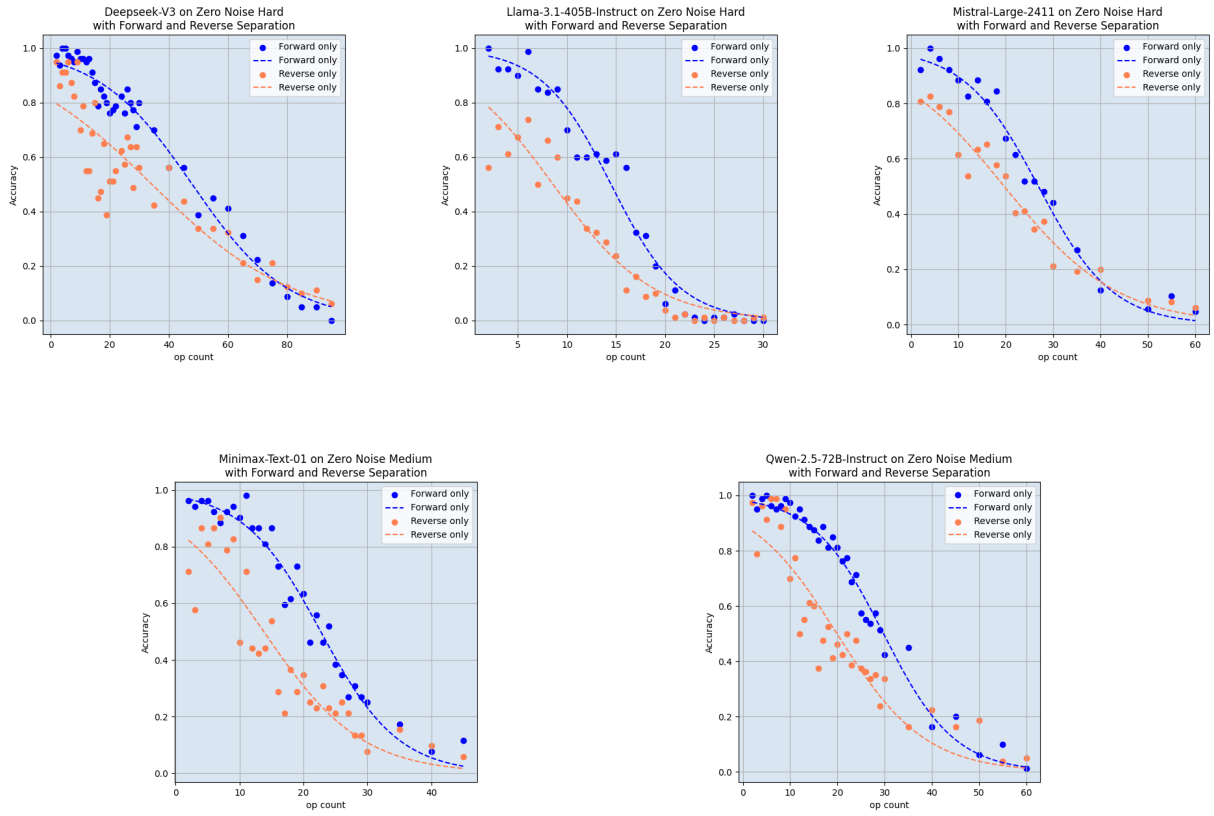

Figure 9. Comparison between forward and reverse

I. Ablation Study of Task Templates

We have three different real-world templates ready. We show that they offer consistent scores with Llama-3.1-8B-Instruct, with slight variables in specific operations. We also show three problem examples.

Template: "Crazy Zootopia"
Problem: The number of adult racoon in South Zoo equals 1 plus the total number of adult animals in Mayer Aquarium. The number of adult fox in Mayer Aquarium equals 2.
Question: What is the total number of adult animals in South Zoo?
Solution: Define adult fox in Mayer Aquarium as t; so t = 2. Define total number of adult animals in Mayer Aquarium as l; so l = t = 2. Define adult racoon in South Zoo as h; n = l = 2; so h = 1 + n = 1 + 2 = 3. Define total number of adult animals in South Zoo as Y; so Y = h = 3. Answer: 3.

Template: "Teachers in School"
Problem: The number of regional medical school in Brightford equals 1. The number of elementary school in Hawkesbury equals 1 plus the total number of schools in Brightford.
Question: What is the total number of schools in Hawkesbury?
Solution: Define regional medical school in Brightford as B; so B = 1. Define total number of schools in Brightford as y; so y = B = 1. Define elementary school in Hawkesbury as T; m = y = 1; so T = 1 + m = 1 + 1 = 2. Define total number of schools in Hawkesbury as q; so q = T = 2. Answer: 2.

Template: "Movie Festival Awards"
Problem: The number of solemn period drama in Festival de Clairmont equals 1 plus the total number of movies in Festival de Saint-Rivage. The number of calm road movie in Festival de Saint-Rivage equals 3.
Question: What is the total number of movies in Festival de Clairmont?
Solution: Define calm road movie in Festival de Saint-Rivage as Z; so Z = 3. Define total number of movies in Festival de Saint-Rivage as x; so x = Z = 3. Define solemn period drama in Festival de Clairmont as e; o = x = 3; so e = 1 + o = 1 + 3 = 4. Define total number of movies in Festival de Clairmont as G; so G = e = 4. Answer: 4.

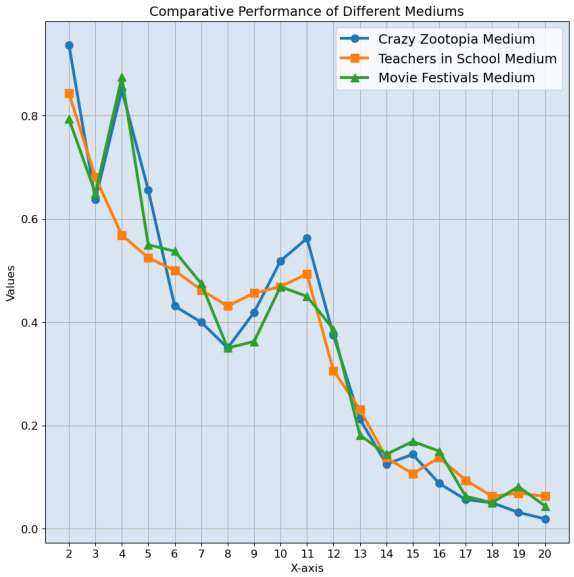

Figure 10. Comparison between different task templates

J. Performance Degradation pattern related to Training Tokens

Previous analyses have demonstrated a sigmoidal relationship between model complexity and performance degradation. To systematically examine how this relationship varies with training data scale, we conducted empirical investigations across different token budgets. Our findings delineate two distinct behavioral regimes: (1) an exponential decay pattern emerges under constrained token budgets, while (2) a characteristic sigmoidal progression manifests in compute-optimal training scenarios. This dichotomy underscores the pivotal role of data sufficiency in determining the degradation dynamics of capacity-scaled models, revealing that scaling laws are fundamentally mediated by the adequacy of training resources.

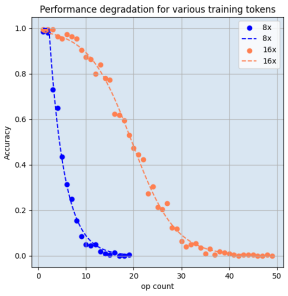

Figure 11. Performance Degradation pattern related to Training Tokens. For constrained token budgets, the curve shows an exponential decay pattern. For more token budgets, the curve shows an sigmoidal decay pattern.

K. Long-Context Degradation of Models

In this section, we provide the accuracy decay curves of all LLMs tested across zero-context, 8K, 16K and 32K for further analysis. We selected the first 30 reasoning steps to truncate the data for comparison purposes.

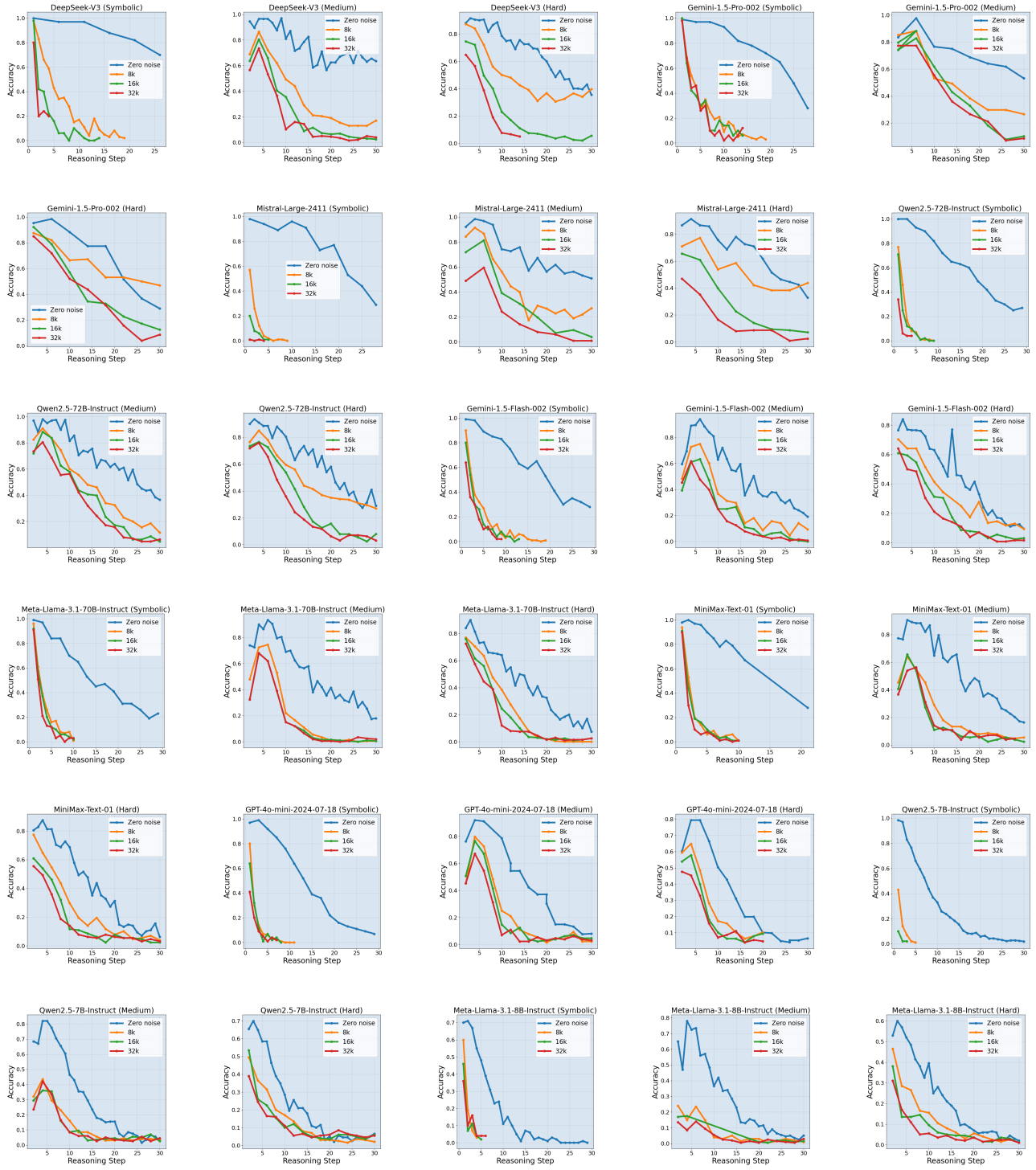

Figure 12. Accuracy decay with context length for different models