



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

Yang Zhou^{1*}, Hongyi Liu^{1*†}, Zhuoming Chen¹, Yuandong Tian², Beidi Chen¹

¹Carnegie Mellon University

²Meta AI

{yangzho6, zhuominc, beidic}@andrew.cmu.edu

liuhongy21@gmail.com, yuandong@meta.com

* Equal contributions. A coin flip decides the order.

† Work done during an internship at Carnegie Mellon University.

Long-context large language models (LLMs) have recently 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.



Github: https://github.com/Infini-AI-Lab/gsm_infinite/

Website: https://infini-ai-lab.github.io/gsm_infinite/

1 Introduction

Recently, state-of-the-art long-context LLMs (Team et al., 2024a; MiniMax et al., 2025) have achieved astonishing performance in a tremendously long context, where Team et al. (2024a) achieves near-perfect performance in 10M multimodal 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.

Although widely used, current long-context bench-

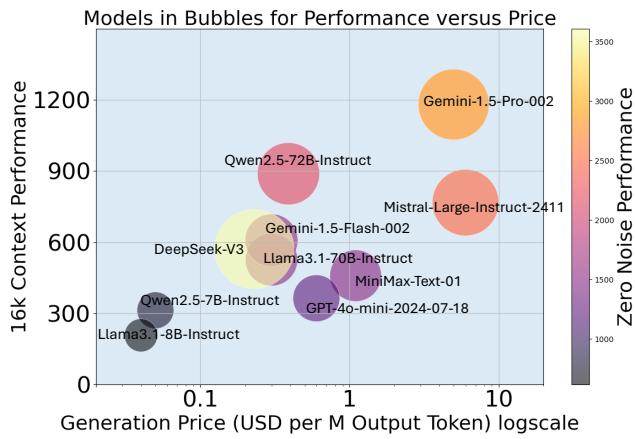


Figure 1 Evaluation of 10 powerful LLMs on GSM-∞, comparing API generation cost (horizontal axis) with zero-context reasoning ability (vertical axis). Bubble size represents reasoning performance at a 16K context length.

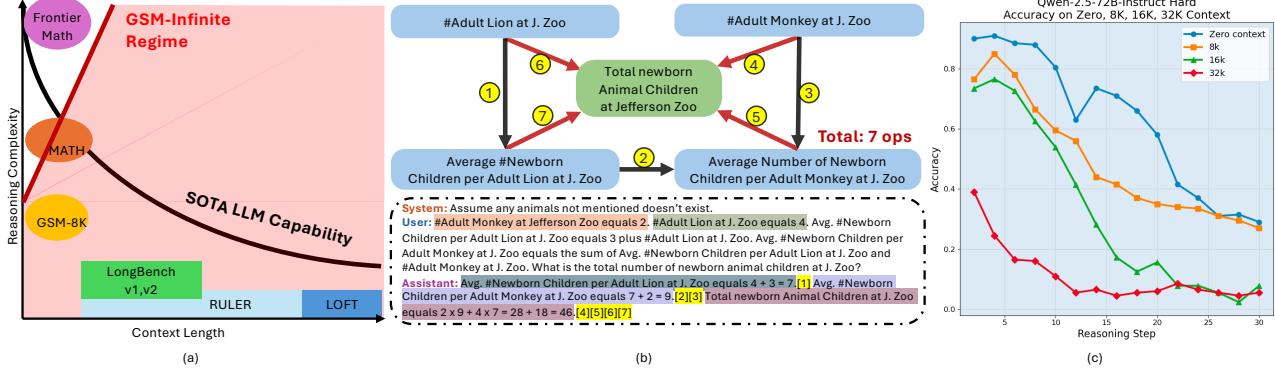


Figure 2 (a) We position existing benchmarks across the Reasoning complexity versus context length plot. Reasoning datasets are usually of very short context. Existing long context benchmarks are usually low in reasoning complexity. Our task can cover any context length that the user so chooses and can 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 computational graph, and we then based on the graph, attach real-world context to it to formulate the problem statements. (c) Shows Qwen2.5-72B-Instruct Score decay across zero-context, 8K, 16K, and 32K.

marks 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 3). (2) **Detectable Noise**. Many 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 with adequate and verified annotation and labeling. This scarcity limits test diversity and fine-grained difficulty assessment, reducing their effectiveness in model evaluation.

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 computational graphs attached 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 and edges upon the core graph, strategically connected to existing nodes without affecting the necessary reasoning steps for solving the tasks. This design enables the generation of arbitrarily long test examples, from which context-level RAG finds it hard to differentiate relevant information from noise.

However, several technical challenges must be addressed to construct a practical benchmark. First, ensuring diverse reasoning patterns and operations - ideally providing a comprehensive coverage of GSM-8K - is crucial to enable fair and thorough evaluation. Second, computational graphs must be effectively translated into natural language that is both human- and LLM-readable, eliminating ambiguity from memorization and ensuring a clear focus on reasoning.

We introduce GSM- ∞ , a long-context benchmarking framework that scales and controls reasoning complexity and noise through fine-grained manipulation of computational graphs, enabling their translation into diverse, human- and LLM-readable problems (Figure 2(b)). Specifically,

- in Section 3.1, 3.2, we discuss how to control and scale reasoning complexity by manipulating computational graphs and how to insert noise.
- Section 4.1 discusses methods to ensure comprehensive coverage of reasoning patterns appeared in GSM-8K.
- Section 4.2 shows ways enabling the computational graphs to LLM-understandable natural language mapping.

Figure 2(a) shows where our benchmark positions among existing benchmarks, demonstrating the effectiveness of

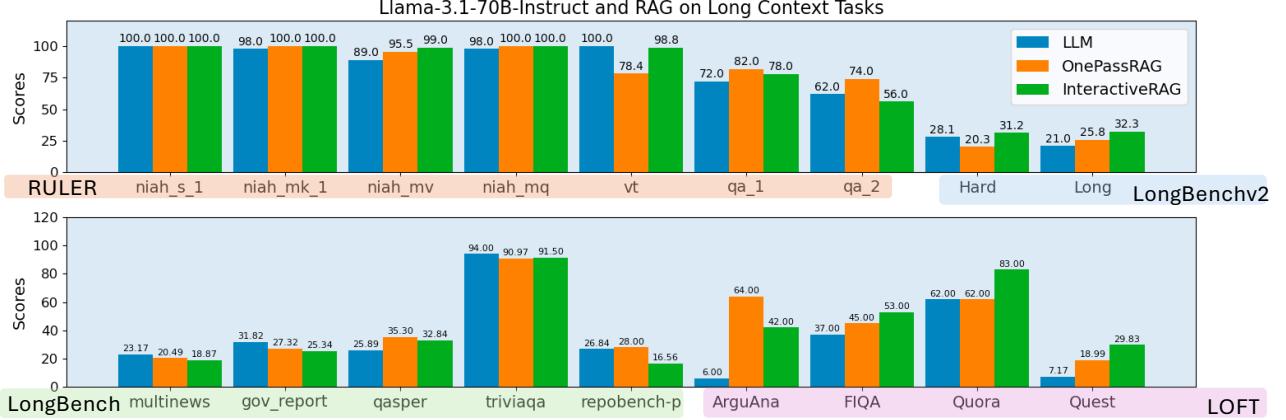


Figure 3 Study of Llama3.1-70B-Instruct with Passive RAG (referred to as OnePassRAG) and Active RAG (referred to as InteractiveRAG) on popular long-context benchmarks: RULER (at 64K context length), LongBench (>8K), LongBenchV2, and LOFT (128K context length). RAG is under the 2048 retrieved token budget, 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.

GSM-∞ for long-context reasoning evaluations.

We conduct a comprehensive evaluation of 17 state-of-the-art LLMs on zero-noise problems and 10 LLMs on various noise-injected tasks using GSM-∞. We inclusively covered a wide range of models, including both popular closed-source and open-source options, conventional transformers and hybrid architectures, models of varying sizes, and both reasoning and non-reasoning LLMs. 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:

- **Decay with reasoning complexity:** LLM performance follows a strikingly consistent sigmoid or exponential decay as problem difficulty increases, highlighting fundamental limitations in scaling reasoning capabilities.
- **Decay with noise:** Performance degradation intensifies as context length increases within the same difficulty level, while longer context brings sharper degradation of the LLM performance.
- **LLM thinking patterns:** Models consistently perform better on forward-thinking tasks than on backward-thinking ones, suggesting a fundamental asymmetry in their reasoning strategies.
- **Influence of Repeated sampling:** Repeated sampling (Brown et al., 2024a) on GSM-∞ reveals a clear pattern: performance improves linearly with increased inference steps but at an exponentially growing computation cost, underscoring inefficiencies in current LLM inference strategies.

2 Related Work and Problem Statement

Despite the wide popularity of some existing long-context benchmarks, this section reveals and elaborates on the three key limitations: low complexity, detectable noise, and low resource or limited quantity of test examples. These three limitations make it extremely challenging to measure long-context LLMs’ progress toward advanced intellectual agents using existing benchmarks.

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. As shown in Figure 3, our experiments demonstrate that RAG systems achieve competitive results with Llama3.1-70B-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

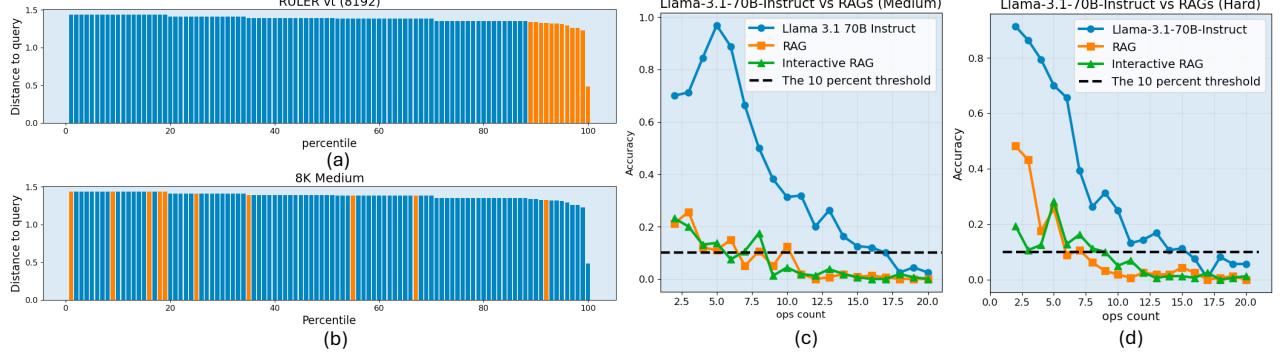


Figure 5 RAG performance on our proposed long-context benchmarks. (a) studies retriever’s behavior on the first 100 chunks of a random problem in vt from RULER with 8192 context length. The chunks that need to be retrieved to solve the problem are labeled in coral, while the noise is in blue. The chunks 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)

tracking (RULER-vt) and code completion (LongBench-repobench-p). 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, Llama3.1-70B-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 5 (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 (Math and coding) heavily rely on human annotation and labeling and have test examples in limited quantity. Here we use GSM-8K (Cobbe et al., 2021a) as an example, shown in Figure 4(a). 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 $op \geq 8$. This scarcity makes meaningful evaluation impractical. Also, DafnyBench (Loughridge et al., 2024), a high-quality long-context coding benchmark only contains 782 verified and deduped examples.

Problem Statement - *How can we develop 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?*

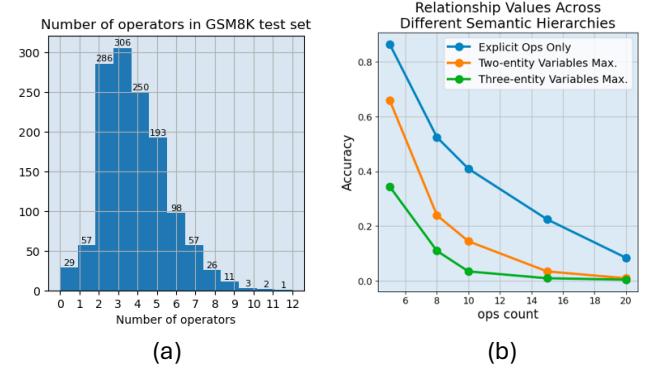


Figure 4 (a) presents a conservative estimate for each problem difficulty in GSM-8K 1.3K test set. We evaluate the difficulty of the problems by the number of operations needed to get to the final answer. The op count ranges from 2 to 12, while most are around 3-4. (b) shows the Llama3.1-8B-Instruct performance across different semantics hierarchies, revealing the hidden reasoning difficulty innate in natural language.

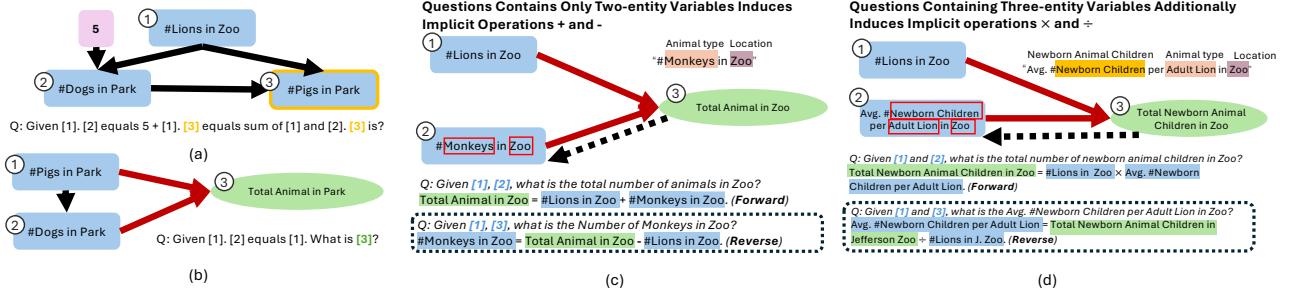


Figure 6 Computational Graphs Illustration. (a) shows a simple computational graph, where every node and edge can be straightforwardly converted to a statement, the other way conversion works as well. (b) breaks down the essence of implicit operations and the abstract parameter constructions; here, an assumption provided in the system prompt is to assume the animals not mentioned in question don’t exist; essentially, problems in natural language omit the two red arrows in the computational graph. (c) provides an example problem when all variables that appeared are “two-entity” variables, where only implicit addition/subtraction can be generated. (d) contrasts (c) and shows an example that with additional “three-entity” variables, the computation graph can also generate implicit multiplication/division. Both (c) and (d) also illustrate the design of reverse mode that specifically aims to generate implicit subtraction and division. (Figure best viewed in color)

3 Computational Graphs

In this section, we detailed the construction of computational graphs, the key construct that enables GSM- ∞ generation of infinite quantities of arbitrary context length and reasoning complexity. Specifically, we explain in detail the potential mapping between reasoning problems and computational graphs in Section 3.1, and in Section 3.2, we propose to generate indistinguishable noise by strategically extending the computational graph.

3.1 Graph Construction to Build Reasoning Problems

After carefully studying GSM-8K problems, we draw the following crucial observations, which allow us to map a randomly generated computational graph to grade-school-level math reasoning problems that cover all possible operations and relationship types.

Mapping Explicit Ops to Computational Graphs - From Every operation used in the GSM-8K is one of the four “+”, “-”, “ \times ”, and “ \div ”. Consider the following example when operations are presented explicitly, “Eggs cost twice as much as tomatoes, while tomatoes cost 1 dollar each.” These statements mention operations (“plus”, “more”, “times”, etc.) can easily be abstracted out as a computational 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”. Another similar example is shown in Figure 6 (a). Therefore, randomly generating a computational 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 Computational 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”. We also illustrate it in Figure 6(b). Following Ye et al. (2024a), we connect the specific instance parameters to the abstract parameters using red edges. To reiterate, these edges are omitted, forcing the LLMs to use commonsense and inducing implicit operations.

Generating Implicit \times using Computational Graphs - every variables in the above-mentioned problem

with the implicit “+” operation are ”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, if every variable in the problem are two-entity” variables, only generate implicit operations of + and - will be induced, but not \times and \div . For a problem to contain implicit operations \times , 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”. We further contrast the above scenario in Figure 6(c) and (d), we see that if every variable is two-entity or ”Animal” in ”Location”, generating \times isn’t straightforward, but once we add in the third entity, or an attribute of animal, generation of \times becomes natural.

Scaling up Reasoning Complexity and maintaining control with Computational Graph - Our computational graph generator employs the abstract parameter construct to generate implicit operations and three-entity variables to represent multiplication operations. During the generation process of a problem, specifically, a query node is first sampled from the graph. 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.

3.2 Noise Construction Using Computational Graphs

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 computational 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 inter-connecting edges between newly added noise nodes won’t contribute help deterring 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 6(c).

We evaluated the resulting noise using two RAG systems presented in 2. The results are shown in Figures 5(c) and (d). Llama3.1-70B-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 5(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).**

4 GSM- ∞

In this section, we present key techniques that enable the synthetic dataset to be diverse in operations in Section 4.1, LLM-understandable, and enable the evaluation to be free from non-reasoning factors in Section 4.2. 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- ∞ detailed in Section 4.3.

4.1 Challenge 1: How to Generate Implicit – and \div Operations?

Firstly, we review why the abstract-instance construct can only generate “+” and \times but not “-” and \div . 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 Figure 6 (a) and (b), the design dictates that the specific and detailed variables should be defined before a more abstract variable.

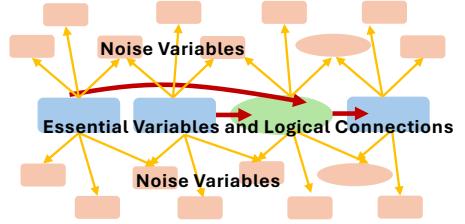


Figure 7 The Illustration of Noise as Extension of Core Computational Graph.

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 implicit ‘-’ operations for 2-entity problems and implicit “ \div ” 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 implicit operations, we introduce a “reverse mode” to generate 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 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 6 (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 are presented in Appendix C.3 of the Appendix.

4.2 Challenge 2: How to Ensure LLM-understandable Problem Generation?

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. Through extended trial and error, variable names with different entity numbers are **presented naturally** in all prepared templates.

Second, the number of constraints applied to the random graph generation process is as little as possible, forcing templates to **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 G 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).

4.3 Benchmark Details

With the synthetic problem generators detailed in Section 4, 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

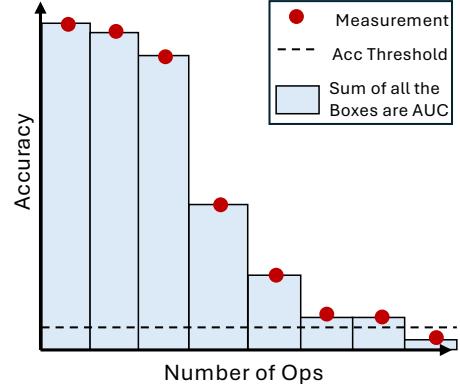


Figure 8 Area Under Curve Metrics is Used to Compare between LLM Performance.

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. Due to space constraint, “Mistral-Large-Instruct-2411” is shortened as “Mistral-Large”; “Claude-3.5-Sonnet” and “Claude-3.5-Haiku” has version number 20241022; “GPT-4o-2024-11-20” is shortened as “GPT-4o” and “GPT-4o-mini-2024-07-18” is shortened as “GPT-4o-mini”.

Models	Three Subtasks			Detailed Statistics on Hard Subtask			Score
	Symbolic	Medium	Hard	1st<50% op	1st<10% op	Avg. Acc op≤30	
DeepSeek-R1	7280.0	9750.85	8573.8	100	>130	0.9427	8534.88
GPT-o3-mini	6690.0	8335.66	5769.96	70	110	0.9423	6931.88
GPT-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	2332.5	2879.92	2310.49	24	50	0.6645	2507.64
Qwen2.5-72B-Instruct	2048.0	2496.81	2016.38	21	40	0.5433	2187.06
GPT-4o	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
Llama3.1-70B-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
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
Qwen2.5-7B-Instruct	786.95	886.75	618.5	7	19	0.2257	764.07
Llama3.1-8B-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

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 used all three templates and mixed the generated problems together to ensure diversity. For reporting LLM performance, we use **Area Under Curve (AUC)**, as shown in Figure 8, which is computing a Riemann sum over the LLM’s performance in accuracy versus the 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.

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
Qwen2.5-72B-Instruct	927.33	681.53	563.65	724.17
Mistral-Large	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
Llama3.1-70B-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
Qwen2.5-7B-Instruct	248.00	211.50	196.17	218.56
Llama3.1-8B-Instruct	183.67	149.50	109.45	147.54

5 Evaluation

In this section, we present comprehensive evaluations of various LLMs on GSM- ∞ . Specifically, the section is organized as follows:

- Section 5.1 presents the complete comparison of LLMs on both zero-noise tasks and long-context tasks. Besides

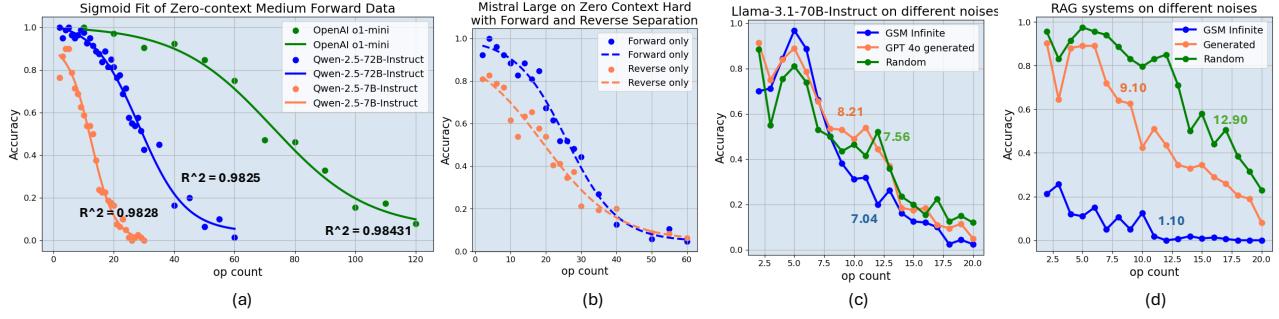


Figure 10 (a) shows three different LLMs’ behavior on GSM- ∞ benchmark zero-context Medium Forward, GPT o1-mini, Qwen2.5-72B-Instruct, and Qwen2.5-7B-Instruct. These models are drastically different in reasoning ability but have performance modeled by sigmoid well, all with $R^2 > 0.98$. (b) shows the 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.

the main leaderboard, we further share four interesting findings.

- Section 5.2 reveals that the sigmoid function generally fits the LLM performance degradation to the increasing ops well.
- Section 5.3 shows that LLMs generally and consistently perform forward problems better than reverse problems. (Defined in 4.1)
- Section 5.4 discuss that various LLM performance decay over longer context and further ablation of the noise.
- Section 5.5 shows that on GSM- ∞ , exponential increase in inference compute yields linear AUC gains.

5.1 Leaderboard

We evaluated 18 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. On the other hand, models with hybrid architecture aren’t performing strong on the zero-noise pure reasoning benchmarks. MiniMax-Text-01 and Jamba both severely underperform compared to 70 B-level models. 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 models. Reasoning model evaluation remains too costly or slow for current long-context leaderboard participation.

Later on, we examine the behavior of LLMs on GSM- ∞ and summarize four interesting findings uniquely enabled by our method of problem generation.

5.2 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 10 for forward problems

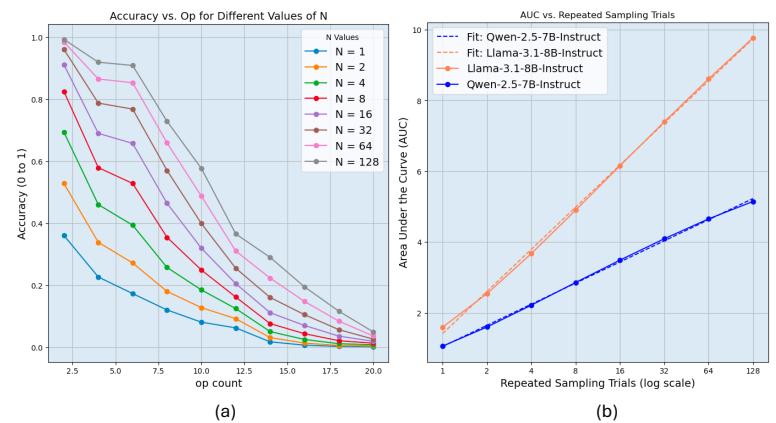


Figure 9 (a) shows repeated sampling on zero-context Hard task with Qwen-2.5-7B-Instruct; (b) shows the AUC to repeated sampling number of trials. We show that for repeated sampling, exponentially increasing inference compute only leads to a linear increase in AUC improvement.

(see Section 4.1 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. Interestingly, while LLMs vary in overall capability, they follow the same trend, differing primarily in the decay rate of the sigmoid function.

5.3 Reverse Problems are Harder to Solve for LLMs

The generator of GSM- ∞ can generate both the "forward" and the "reverse" problems, which can be compared separately. Most LLMs perform worse in reverse problems than forward ones, shown in Figure 10(b) using Mistral-Large (Jiang et al., 2023a). For the Medium, only Jamba-1.5-Large out of a total of 18 LLMs evaluated do reverse problems better than forward problems, and the average difference in AUC is 604. For the Hard subtask, 5 out of 18 models have larger reverse problems AUC, Deepseek R1, Gemini-1.5-Pro-002, Qwen2.5-7B-Instruct (Qwen et al. (2025), 4o-mini, and Jamba-1.5-Large (Team et al., 2024b)). The average difference in AUC is 154. A detailed breakdown is listed in Appendix F. Besides, LLM performance on reverse problems can also be modeled by a sigmoid mapping. Five more LLM plots are presented in Appendix F.

5.4 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 H.

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 Llama3.1-70B-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.

5.5 Limitations of Repeated Sampling

The construction of GSM- ∞ allows us to study techniques for Inference Scaling as well. Specifically, we study repeated sampling (Brown et al., 2024a; Snell et al., 2024) and its effectiveness under different reasoning complexity. We study both Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct with the best-of-N settings similar to Brown et al. (2024b). 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, Qwen2.5-7B-Instruct behavior is plotted in Figure 9(a) with different repeated trial settings.

Surprisingly, suppose we calculate the AUC score under every curve corresponding to each number of repeated trial settings. In that case, we find that the increment in the AUC score from two consecutive settings is close. If we 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 9(b). The R-squared is greater than 0.99 for both Qwen2.5-72B-Instruct and Llama3.1-8B-Instruct, where Llama3.1-8B-Instruct. **Therefore, GSM- ∞ helps reveal that Repeated Sampling leads to linear AUC Improvement from exponentially increasing computation cost.**

6 Conclusion

Long-context LLMs have the potential to tackle complex, information-dense tasks requiring deep reasoning and coherent long-form generation. 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

- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. <https://arxiv.org/abs/2308.14508>.
- Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks, 2025. <https://arxiv.org/abs/2412.15204>.
- Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. <https://arxiv.org/abs/2004.05150>.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024a.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024b. <https://arxiv.org/abs/2407.21787>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021a. <https://arxiv.org/abs/2110.14168>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021b.
- Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhua Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoping Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyu Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. <https://arxiv.org/abs/2501.12948>.
- Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural networks and the chomsky hierarchy, 2023. <https://arxiv.org/abs/2207.02098>.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Bin Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra,

Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenber, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poult, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghatham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan

Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vitor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. <https://arxiv.org/abs/2407.21783>.

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

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

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024a.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024b. <https://arxiv.org/abs/2404.06654>.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023a.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023b. <https://arxiv.org/abs/2305.06983>.

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

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack, 2024. <https://arxiv.org/abs/2406.10149>.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien M. R. Arnold, Vincent Perot, Siddharth Dalmia, Hexiang Hu, Xudong Lin, Panupong Pasupat, Aida Amini, Jeremy R. Cole, Sebastian Riedel, Iftekhar Naim, Ming-Wei Chang, and Kelvin Guu. Can long-context language models subsume retrieval, rag, sql, and more?, 2024. <https://arxiv.org/abs/2406.13121>.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models, 2024. <https://arxiv.org/abs/2402.14848>.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. <https://arxiv.org/abs/2005.11401>.

Xinze Li, Yixin Cao, Yubo Ma, and Aixin Sun. Long context vs. rag for llms: An evaluation and revisits. *arXiv preprint arXiv:2501.01880*, 2024a.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 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*, pages 881–893, 2024b.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023a. <https://arxiv.org/abs/2310.01889>.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023b. <https://arxiv.org/abs/2307.03172>.

Chloe Loughridge, Qinyi Sun, Seth Ahrenbach, Federico Cassano, Chuyue Sun, Ying Sheng, Anish Mudide, Md Rakib Hosain Misu, Nada Amin, and Max Tegmark. Dafnybench: A benchmark for formal software verification, 2024. <https://arxiv.org/abs/2406.08467>.

MiniMax, Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, Enwei Jiao, Gengxin Li, Guojun Zhang, Haohai Sun, Houze Dong, Jiadai Zhu, Jiaqi Zhuang, Jiayuan Song, Jin Zhu, Jingtao Han, Jingyang Li, Junbin Xie, Junhao Xu, Junjie Yan, Kaishun Zhang, Kecheng Xiao, Kexi Kang, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Zheng, Limbo Chai, Long Xing, Meizhi Ju, Mingyuan Chi, Mozhi Zhang, Peikai Huang, Pengcheng Niu, Pengfei Li, Pengyu Zhao, Qi Yang, Qidi Xu, Qiexiang Wang, Qin Wang, Qiuwei

Li, Ruitao Leng, Shengmin Shi, Shuqi Yu, Sichen Li, Songquan Zhu, Tao Huang, Tianrun Liang, Weigao Sun, Weixuan Sun, Weiyu Cheng, Wenkai Li, Xiangjun Song, Xiao Su, Xiaodong Han, Xinjie Zhang, Xinzhu Hou, Xu Min, Xun Zou, Xuyang Shen, Yan Gong, Yingjie Zhu, Yipeng Zhou, Yiran Zhong, Yongyi Hu, Yuanxiang Fan, Yue Yu, Yufeng Yang, Yuhan Li, Yunan Huang, Yunji Li, Yunpeng Huang, Yunzhi Xu, Yuxin Mao, Zehan Li, Zekang Li, Zewei Tao, Zewen Ying, Zhaoyang Cong, Zhen Qin, Zhenhua Fan, Zhihang Yu, Zhuo Jiang, and Zijia Wu. Minimax-01: Scaling foundation models with lightning attention, 2025. <https://arxiv.org/abs/2501.08313>.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

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

Vasudev Shyam, Jonathan Pilault, Emily Shepperd, Quentin Anthony, and Beren Millidge. Tree attention: Topology-aware decoding for long-context attention on gpu clusters, 2024. <https://arxiv.org/abs/2408.04093>.

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

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024a.

Jamba Team, Barak Lenz, Alan Arazi, Amir Bergman, Avshalom Manovich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, et al. Jamba-1.5: Hybrid transformer-mamba models at scale. *arXiv preprint arXiv:2408.12570*, 2024b.

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

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process, 2024a. <https://arxiv.org/abs/2407.20311>.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems, 2024b. <https://arxiv.org/abs/2408.16293>.

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

Tan Yu, Anbang Xu, and Rama Akkiraju. In defense of rag in the era of long-context language models, 2024b. <https://arxiv.org/abs/2409.01666>.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞ bench: Extending long context evaluation beyond 100k tokens, 2024. <https://arxiv.org/abs/2402.13718>.

Appendix Table of Contents

We present the following table of contents to better traverse the Appendix.

- Related Work
 - Long-context Language Models
 - Long context benchmarks and tasks
 - Limitation of Existing Reasoning Tasks
 - Synthesized Datasets for Long-context
- Benchmark Limitation
- Detailed Experiment Setup
- Full Result of RAG Experiments
- Illustrative Problems
- Forward and Reverse Problems Breakdown
- Ablation Study of Task Templates
- Long-Context Degradation of Models

A Related Work

A.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).

A.2 Long context benchmarks and tasks

There have been 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.

A.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 4(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 $op \geq 9$, 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 and MATH problems since MATH uses operations such as taking power or roots 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 2 that the addition of irrelevant noise cannot meaningfully evaluate the ability to reason in a long context of LLMs.

A.4 Synthesized Datasets for long-context

Synthesized tasks are simple to build and absolutely deterministic, data contamination safe, but highly effective in evaluating 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. (2021b) 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 reasoning benchmarks for the long-context community.

B Benchmark Limitation

Despite our careful construction and attention to LLM understandability, there are still several important limitations to the existing benchmark GSM- ∞ . First, our current benchmark only targets operations contained in the grades school math, so only $+, -, \times, \div$ are included. We are aware that Math (Hendrycks et al., 2021) includes more complex operators, which we are looking at incorporating into the next version of the benchmark. The key limitation it brings to our benchmark is shown in Figure 2(a), to generate complex problems with high reasoning complexity, our generator needs to generate problems that are much longer than hard mathematic benchmarks, despite our limitless quantity and complexity scalability. Second, another key limitation of synthetic benchmarks is the lack of diversity in natural language. Natural Language reasoning tasks usually contain multiple ways to assign a variable value or to describe a relationship between multiple variables, which is extremely difficult to program in software without the LLMs in the loop. We deliberately aim to keep LLMs out of the loop for better scalability and counter the diversity by incorporating more than one template into the problem generation. Third, we implement multiple checks to ensure all of the essential steps in the core graph have positive values and have a number range of less than four figures, but for the current noise generation, we don’t implement these checks, since they are redundant to the problems. However, we do notice that when processing through the noise, the LLMs sometimes are confused by the irregular noise variable values. In later version of the benchmark, we will address this issue and make sure the entire graph is consistent.

C Detailed Experiment Setup

C.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.

C.2 Repeated Sampling Experiment Setup

C.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:

$$p_{\text{task}} = \frac{\# \text{ Correct Samples}}{256} \quad (1)$$

- Estimate accuracy for N samples:

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

3. Aggregation

- Average results across 80 tasks:

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

C.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

C.3 Some Small Implementation Tips to Avoid Quadratic (or higher-order) Equations and Multiple Possible Solutions

The quadratic equation gives rise to the following particular situation. For solving reverse mode questions, one easy way is to assign the variable asked as X, perform symbolic operations carrying with X, and go through the entire computation graph to close the circle and formulate an equation. Using that equation would easily solve for X. However, sometimes the query variable is too deep in the chain, and it might assign both variables Y and Z, which are later multiplied. Then, two linear expressions are multiplied to get quadratic operations, and as op becomes larger, it can be more severe than quadratic.

To alleviate the situation, we found that the following technique is empirically helpful. When generating the problem statement, we first locate the variable that has the largest number of “entities” in the variable name and

make sure that that one is the end of the topology sort list during the forward mode generation. Then, when switching to reverse mode, we locate the last possible initialized variable, so as close to the end of the sort list as possible, and assign this variable as a query, effectively reducing the chance of its dependents multiplying each other. The method can reduce the chance of quadratic or higher-order equations happening. However, if we still encounter the cases even when the above procedure is in place, we discard the generated example and ask the generator to generate again.

D Full Result of RAG experiments

Here, we present the full result of the RAG experiment for further analysis. We retrieve 2048 tokens for each RAG retrieval.

D.1 RULER

Models	s1	s2	s3	mk1	mk2	mk3	mv	mq	Context Length
Llama3.1-70B-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
Llama3.1-70B-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
Llama3.1-70B-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
Llama3.1-70B-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
Llama3.1-70B-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
Llama3.1-70B-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)

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

D.2 LongBench V2

Tasks	Overall	Easy	Hard	Short	Long
Llama3.1-70B-Instruct	30	33.3	28.1	44.7	21
OnePassRAG	25	33.3	20.3	23.7	25.8
InteractiveRAG	33	36.1	31.2	34.2	32.3

Table 5 RAG vs Model (LongBench V2)

D.3 LongBench

Tasks	passage_count	hotpot-qa	samsum
Llama3.1-70B-Instruct	36.0,36.0,32.0	58.87,71.22,76.44	28.88,35.95,41.48
OnePassRAG	0.0,0.0,0.0	65.04,61.59,63.05	31.63,23.28,26.84
InteractiveRAG	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
Llama3.1-70B-Instruct	27.71,24.81,23.17	57.31,51.83,64.98	34.94,34.97,31.82
OnePassRAG	26.85,22.72,20.49	51.69,48.66,55.85	32.6,30.67,27.32
InteractiveRAG	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
Llama3.1-70B-Instruct	50.3,46.5,25.89	100.0,100.0,100.0	74.93,64.37,59.6
OnePassRAG	45.73,43.5,35.3	72.0,72.0,79.0	67.97,59.64,48.4
InteractiveRAG	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
Llama3.1-70B-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	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	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

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

D.4 LOFT

Tasks	ArguAna	FEVER	FIQA	MS MARCO	NQ	Quora	SciFact
Llama3.1-70B-Instruct	0.06	0.78	0.37	0.67	0.84	0.62	0.59
OnePassRAG	0.64	0.88	0.45	0.77	0.86	0.62	0.64
InteractiveRAG	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
Llama3.1-70B-Instruct	0.4411	0.37	0.2	0.024	0.07166
OnePassRAG	0.2529	0.455	0.2383	0.1559	0.1899
InteractiveRAG	0.79	0.29	0.13	0.1539	0.2983

Table 11 RAG vs Model (LOFT) - Part 2

Comments: Due to LOFT’s limited prompt availability (no official releases for 32k/1M contexts), we conducted experiments solely on 128k context lengths. We observed that LOFT’s document ID retrieval tasks inherently require document identifiers that aren’t captured in standard RAG-retrieved chunks. To address this, we implemented a lightweight modification: appending document ID tags to each context chunk during retrieval. This allows the RAG system to infer the correct document ID when retrieving relevant content, resolving the task-specific limitation without altering core RAG functionality.

E 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

Three example problems one for each subtask	
Problem	<ul style="list-style-type: none">• 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.\nUsing 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 South Zoo 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.

F Forward and Reverse Problems Breakdown

Models	Forward Problem	Reverse Problem	Forward AUC - Reverse AUC
Llama3.1-70B-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
Llama3.1-8B-Instruct	1030.000000	563.125000	466.875000
Claude-3.5-Sonnet	3653.830050	3158.657850	495.172200
Qwen2.5-72B-Instruct	2889.375000	2141.250000	748.125000
Qwen2.5-7B-Instruct	995.625000	833.125000	162.500000
o1-mini	6517.510550	5592.307100	925.203450
Gemini-1.5-Flash-002	1889.375000	1153.750000	735.625000
Claude-3.5-Haiku	1234.620000	873.100000	361.520000
Llama-3.1-405B-Instruct	1781.400000	981.250000	800.150000
DeepSeek-V3	4613.125000	3713.125000	900.000000
Gemini-1.5-Pro-002	4204.564075	3160.574950	1043.989125
DeepSeek-R1	9764.950000	9750.950000	14.000000
MiniMax-Text-01	2148.071300	1539.415650	608.655650
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-3.5-Haiku	819.240000	776.900000	42.340000
Llama3.1-70B-Instruct	1314.375000	1098.750000	215.625000
Gemini-1.5-Flash-002	1341.250000	1219.375000	121.875000
MiniMax-Text-01	1360.555000	1034.625000	325.930000
DeepSeek-R1	8444.500000	8756.950000	-312.450000
o1-mini	3831.381000	3645.474200	185.906800
Gemini-1.5-Pro-002	2255.732025	2444.270375	-188.538350
DeepSeek-V3	2725.085000	2109.560000	615.525000
Qwen2.5-7B-Instruct	625.625000	630.625000	-5.000000
GPT-4o	1592.280000	1311.560000	280.720000
Llama3.1-8B-Instruct	759.375000	460.625000	298.750000
Qwen2.5-72B-Instruct	2196.875000	1895.000000	301.875000
Claude-3.5-Sonnet	2242.309950	1999.998100	242.311850
GPT-4o-mini	858.400000	873.310000	-14.910000
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

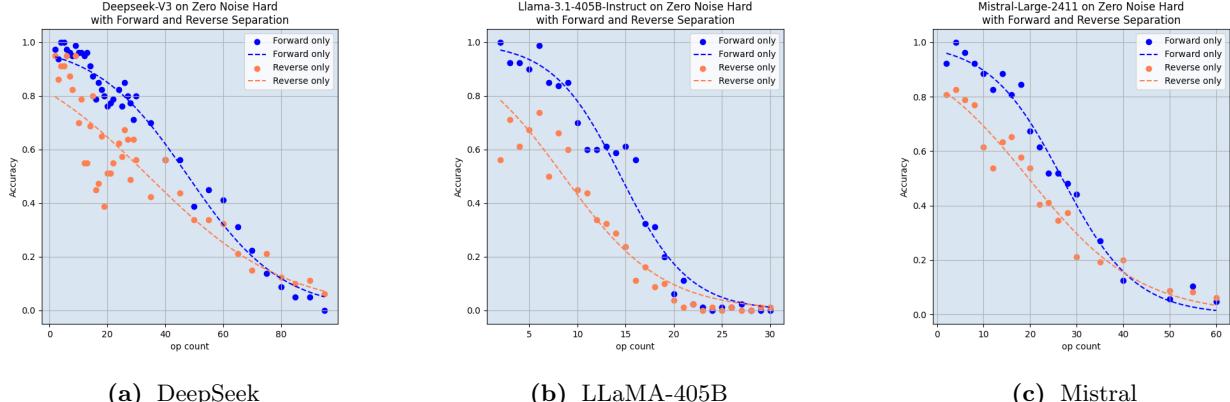


Figure 11 Comparison of different models

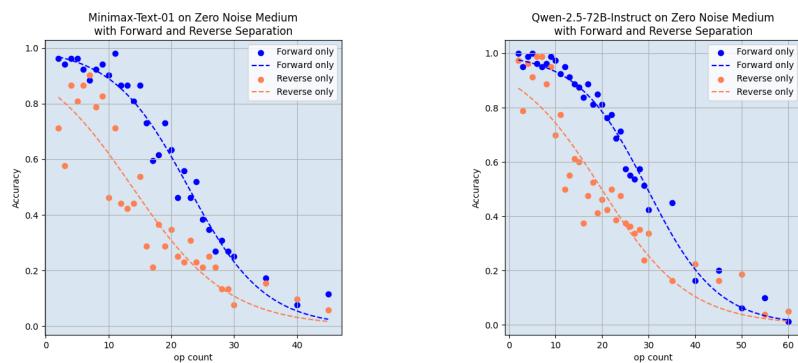


Figure 12 Comparison between forward and reverse

G 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.

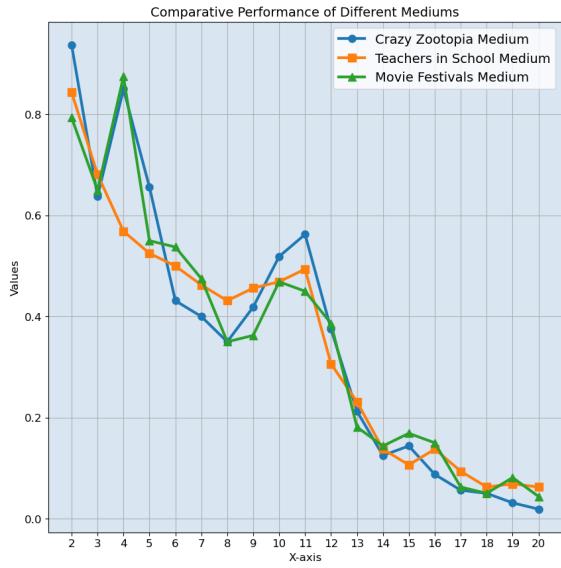


Figure 13 Comparison between different task templates

H 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.

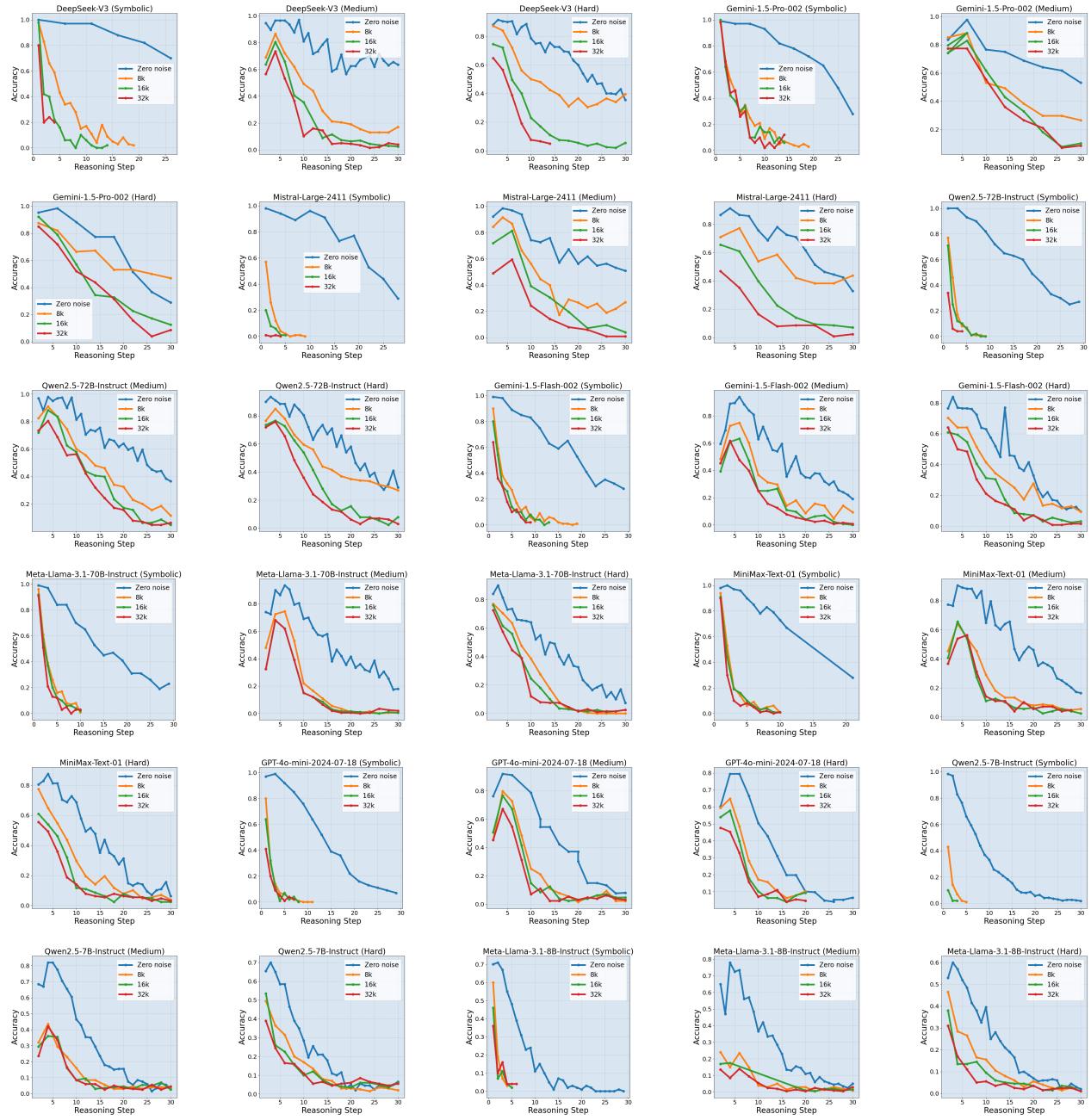


Figure 14 Accuracy decay with context length for different models