LongGenBench: Long-context Generation Benchmark

Xiang LIU **Peijie DONG** **Xuming HU**† **Xiaowen CHU**†
The Hong Kong University of Science and Technology(Guangzhou)
{xliu886,pdong212}@connect.hkust-gz.edu.cn
xuminghu@hkust-gz.edu.cn xwchu@ust.hk

Abstract

Current long-context benchmarks primarily focus on retrieval-based tests, requiring Large Language Models (LLMs) to locate specific information within extensive input contexts, such as the needle-in-a-haystack (NIAH) benchmark. Long-context generation refers to the ability of a language model to generate coherent and contextually accurate text that spans across lengthy passages or documents. While recent studies show strong performance on NIAH and other retrieval-based long-context benchmarks, there is a significant lack of benchmarks for evaluating long-context generation capabilities. To bridge this gap and offer a comprehensive assessment, we introduce a synthetic benchmark, **LongGenBench**, which is designed to evaluate the long-context generation capabilities of large language models (LLMs), with a particular focus on consistency in logical flow. LongGenBench redesigning the format of questions and necessitating that LLMs respond with a single, cohesive long-context answer. Upon extensive evaluation using LongGenBench, we observe that: (1) both API accessed and open source models exhibit performance degradation in long-context generation scenarios, ranging from 1.2% to 47.1%; (2) different series of LLMs exhibit varying trends of performance degradation, with the Gemini-1.5-Flash model showing the least degradation among API accessed models, and the Qwen2 series exhibiting the least degradation in LongGenBench among open source models.

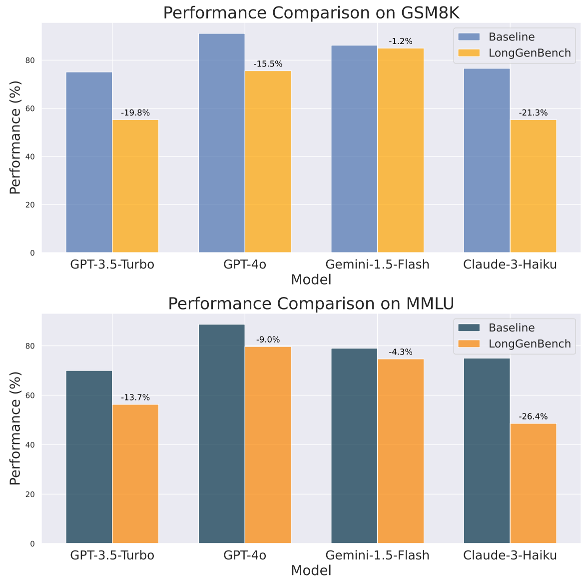

Figure 1: Performance Comparison of LLMs on GSM8K and MMLU datasets using LongGenBench to assess their long-context generation capabilities. It is observed that mainstream LLMs exhibit performance degradation when tasked with long-context generation.

1 Introduction

[1] Large Language Models (LLMs) have become pivotal in tackling NLP downstream tasks such as summarization and question answering that require interpreting extensive context from books, reports, and documents, sometimes spanning tens of thousands of tokens (Raffel et al., 2020; Brown et al., 2020; Chowdhery et al., 2022; Tay et al., 2022; Touvron et al., 2023; Tang et al., 2023; Zhang et al., 2023). Recent advances in long-context technology in ML system field (Dao et al., 2022; Dao, 2024; Jacobs et al., 2023; Xiao et al., 2024) and model architecture design (Chen et al., 2023a; Xiong et al., 2023; Chen et al., 2023b; Peng et al., 2024; Dong et al., 2024) have significantly improved the ability of LLMs to process increasingly large input context lengths (Liu et al., 2024a; Young et al., 2024), such as Gemini-1.5-pro model can handle the 1,500-page document (Reid et al., 2024). Although previous studies (AI21, 2024; X.AI, 2024; Reid et al., 2024; Anthropic, 2024; DeepSeek-AI, 2024) often employ synthetic tasks like passkey retrieval (Mohtashami and Jaggi, 2023) and needle-

[1]† Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 865–883
November 12-16, 2024 ©2024 Association for Computational Linguistics

in-a-haystack (NIAH) (Kamradt, 2023) to evaluate the long-context capability of these LLMs, such tasks primarily test retrieval skills and do not fully assess other aspects of long-context generation.

Long-context generation refers to the ability of a language model to generate coherent and contextually accurate text that spans across a lengthy passage or document. This capability involves maintaining the thematic continuity, logical flow, and consistency of details over extended sequences of text, which can include multiple paragraphs, pages, or even entire documents.

To facilitate further research in this area, we propose the **Long**-context **Gen**eration **bench**mark (**LongGenBench**), a new benchmark specifically designed to evaluate the long-context generation capabilities of LLMs, with a particular focus on consistency in logical flow. LongGenBench synthesizes a dataset from current popular LLM benchmarks, redesigns the input format, and includes multiple questions within a single query. The LongGenBench requires LLMs to generate a comprehensive long-context response that sequentially addresses each question. To achieve better performance in LongGenBench, LLMs need to maintain consistency regardless of whether the previous generation part is correct or incorrect. In LongGenBench, evaluating the quality of these long-context responses is straightforward: simply compare the generated answers with the ground truth.

Our study evaluates the performance of various language models using the LongGenBench approach across different datasets, specifically *LongGenBench-MMLU*, *LongGenBench-GSM8K*, and *LongGenBench-CSQA*. Figure 1 displays the performance of four powerful API accessed models tested in both the baseline scenario, which involves single-answer generation, and the LongGenBench scenario, which focuses on long-context generation. It is notable that the Gemini-1.5-Flash model exhibits the lowest performance degradation in long-context generation tasks, surpassing the GPT-4o. Additionally, we conducted analyses on open-source models, revealing a general correlation between baseline performance and LongGenBench performance. **Models with higher baseline scores tend to show smaller declines in long-context generation tasks.** Models like Qwen2-72B-Instruct and DeepSeek-v2-Chat, both with high baseline scores, also exhibit minimal performance degradation. However, there are exceptions, such as LLaMA-3-70B-Instruct, which, despite its high baseline performance, experiences significant performance drops. Moreover, **model size influences performance**, as larger models within the same series, such as the LLaMA-3 and Qwen2 series, demonstrate smaller declines. **Different architectures show varying trends** in performance degradation; for example, LLaMA-3-8B-Instruct shows a performance degradation of 47.1% on GSM8K, while ChatGLM4-9B-Chat only experiences a 10.8% drop, despite having similar baseline performances. **Consistency across tasks** is observed, with models like LLaMA-3-8B-Instruct consistently showing significant drops on all datasets, whereas models such as Qwen2-72B-Instruct and DeepSeek-v2-Chat maintain minimal declines across all datasets, underscoring their resilience in long-context generation tasks. These findings highlight the varying capabilities of different models to maintain accuracy over extended text generation and provide valuable insights for future model development and optimization.

Our contributions are as follows:

- We introduce LongGenBench, an effective approach for evaluating the long-context generation capabilities of language models across multiple datasets.

- We provide a comprehensive performance comparison between API accessed and open source models under the LongGenBench framework, revealing insights into how different models handle long-context generation tasks.

- Our analysis uncovers critical relationships in long-context generation tasks, including the correlation between baseline performance and LongGenBench performance, the impact of model size on performance decline, and the variation among different model architectures. Our detailed experiments establish consistent trends in performance degradation across different LongGenBench tasks, highlighting the importance of model resilience in long-context generation.

2 Related Work

2.1 Long-context Language Models

Recent advancements in techniques such as efficient attention, long-term memory, extrapolative positional embedding, and context processing have spurred the development of numerous long-context LLMs (Huang et al., 2023). Efficient attention

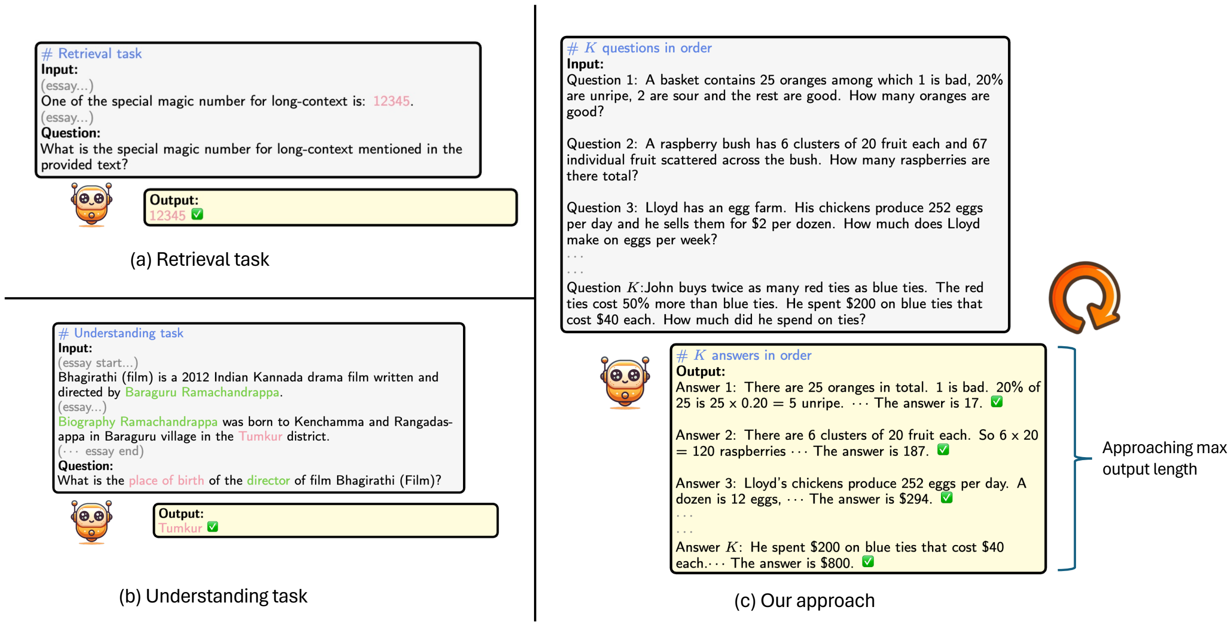

Figure 2: Illustrations of previous long-context benchmarks and our proposed approach. **(a) Retrieval task**: requires LLMs to retrieve the magic information hidden within an unrelated long context. **(b) Understanding task**: requires LLMs to comprehensively understand a long essay and answer the specific question. **(c) Our approach**: reconstructs the format of the dataset, requiring LLMs to sequentially understand and respond to each question in a single response. We run multiple iterations with different questions to evaluate the robustness of long-context generation capabilities. The length of the generated responses aims to approach the token limit.

mechanisms like Flash attention (Dao et al., 2022; Dao, 2024) and Ring attention (Liu et al., 2023) have dramatically reduced memory demands for processing extensive contexts. Moreover, sparse attention methods, including shifted sparse attention in LongLoRA (Chen et al., 2023b), dilated attention (Ding et al., 2023), and attention sinks (Han et al., 2023; Xiao et al., 2024), further enhance long-context capabilities. For long-term memory, efficiency is achieved by caching previous contexts using recurrent mechanisms (Zhang et al., 2024b; Bulatov et al., 2023; Martins et al., 2022; Wu et al., 2022; Mohtashami and Jaggi, 2023). Techniques for extrapolative positional embedding include ALiBi (Press et al., 2022), xPOS (Sun et al., 2023), and RoPE (Su et al., 2024), along with their variants (Chen et al., 2023a; Xiong et al., 2023; Peng et al., 2024; Liu et al., 2024b; Ding et al., 2024; Zhu et al., 2023). In terms of context processing, key information is retained through retrieval augmentation (Xu et al., 2023; Wang et al., 2023; Tworkowski et al., 2024) and prompt compression (Jiang et al., 2023). Innovative architectural designs such as state-space models (Gu et al., 2022; Fu et al., 2022; Poli et al., 2023; Fu et al., 2023a; Gu and Dao, 2023; Dao and Gu, 2024) and RWKV (Peng et al., 2023) are also being developed to effectively manage long-context inputs.

2.2 Evaluation for Long-context Language Models

Numerous investigations into long-context model benchmarks have primarily focused on retrieval and understanding tasks. In the realm of retrieval benchmarks, the datasets used are predominantly synthetic, enabling precise control over experimental conditions, such as input token length, and minimizing the influence of varied parametric knowledge from different training strategies. Recent research has extensively focused on synthetic tasks designed for retrieval (Kamradt, 2023; Mohtashami and Jaggi, 2023; Li et al., 2023; Liu et al., 2024c; Hsieh et al., 2024; Hu et al., 2024a,b; Zhang et al., 2024a), with additional studies exploring the use of long contexts for various types of reasoning (Tay et al., 2021). For understanding benchmarks, LongBench (Bai et al., 2023b) includes evaluations in a bilingual context, covering long-document question answering, summarization, and code completion tasks. ZeroSCROLLS (Shaham et al., 2023) and L-Eval (An et al., 2023) assess a wide array of realistic natural language tasks, such as long-document question answering and query-driven summarization. ∞-Bench (Zhang et al., 2024c) offers challenges that involve content spanning more than 100,000 tokens.

3 LongGenBench

We propose LongGenBench, a synthetic benchmark that is an efficient, low-cost approach focused on evaluating long-context generation in LLMs.

3.1 Motivation

Traditionally, evaluating LLMs for long-context scenarios involves inputting lengthy essays into the models, followed by either retrieval or comprehension questions as depicted in Figure 2(a) and (b). The token length of these essays typically ranges from {4K, 8K, 16K, 32K, 64K, 128K}, with advanced long-context LLMs like Gemini-1.5 (Reid et al., 2024) being tested up to 1M tokens. However, these benchmarks tend to focus predominantly on the prompt tokens or input content, often neglecting the completion tokens or output content and the evaluation of performance regarding these aspects. Furthermore, traditional long-context benchmarks such as the NIAH test are costly, with a 128K NIAH test consuming 8M tokens.

Algorithm 1 Pipeline of **LongGenBench**

Require: System Prompt S, Questions Q, Number of Questions K, Number of Iterations T, Language Model LLM
Ensure: Long-Context Responses R

$R \leftarrow \emptyset$
for $t \leftarrow 0$ to $T-1$ **do**
  $Q_t \leftarrow Q[t \times K : (t+1) \times K]$
  $InputPrompt \leftarrow S +$ concatenate(Q_t)
  $Response \leftarrow LM$.gen$(InputPrompt)$
  $ParsedResponse \leftarrow$ parse$(Response)$
  $R \leftarrow R \cup \{ParsedResponse\}$
  verify$(ParsedResponse, Q_t)$
end for
return R

3.2 Problem Definition

In LongGenBench, the initial step involves redesigning the input prompt format to enable LLMs to generate long-context responses as illustrated in Figure 2(c). We refine the system prompt and restructure the question format so that K questions are sequentially concatenated after the system prompt. Subsequently, the LLMs are expected to adhere to this redesigned prompt and produce a coherent long-context response that answers all K questions. These responses are then parsed to verify the answers to the K questions, where the LLMs must maintain both the sequence and accuracy to demonstrate improved performance in LongGenBench. This process is repeated for T iterations to assess the robustness of the LLMs' long-context generation capabilities at each length, with each iteration featuring unique questions.

The Algorithm 1 gives a pseudocode outline for the LongGenBench. The system prompt S contains instructional information, while Q is a list of questions from the original dataset. For each iteration t, a batch of K questions, Q_t, is selected from Q within the range $[t \times K : (t+1) \times K]$. The selected questions are concatenated to the system S to form the $InputPrompt$. The language model LLM generates a long-context response for the given $InputPrompt$. The response is added to the response set R, then parsed and verified for correctness and sequence. This process is repeated for T iterations, with each iteration featuring a unique set of questions. The final output is the set of long-context responses R.

During the generation process, LLMs may accumulate correct or incorrect reasoning steps, which fall within the scope of LongGenBench's evaluation. These models might generate errors during a single long-context session, and earlier mistakes can influence subsequent outputs. Assessing a model's performance in generating long texts involves evaluating how effectively it manages and mitigates these accumulated errors, and maintains consistency in logical flow. LongGenBench addresses this challenge by requiring models to handle and correct the impact of previous mistakes within a single long-context generation.

The conditional probability that the LLM generates the next token, given the prompt and the previously generated outputs, can be represented as:

$$P(x_{i+1} \mid \textit{InputPrompt}, x_1, x_2, \ldots, x_i)$$

Where $x_1, x_2, \ldots, x_i$ are the tokens generated in LongGenBench, the LLMs are required to produce the output based on the *InputPrompt* and all previously generated tokens.

3.3 Dataset Construction

LongGenBench synthesizes three datasets from different domains: *World Knowledge* from MMLU (Hendrycks et al., 2021), *Arithmetic* from GSM8K (Cobbe et al., 2021), and *Commonsense Reasoning* from CommonSenseQA (Talmor et al., 2019). The MMLU dataset measures a model's ability to understand and reason across 57 diverse categories, using accuracy as the primary evaluation metric. The GSM8K dataset evaluates arithmetic problem-solving skills through 8,000 grade-school level math word problems, using the solving rate as the main metric. CommonSenseQA tests commonsense reasoning with multiple-choice questions based on ConceptNet, with accuracy as the evaluation metric. Appendix A.3 provides details on how the synthesis process occurs.

4 Expeirments Setting

In this section, we describe the details of the baseline models and the LongGenBench approach, as well as their implementation in the subsequent experiments. All experiments were conducted three times, using the mean score to ensure robustness.

Model	Access Method	Context Length	Max Output Length	Input/Output Price
GPT-3.5-Turbo	API	16K	4K	\$0.5 /\$1.5
GPT-4o	API	128K	4K	\$5 /\$15
Gemini-1.5-Flash	API	1024K	8K	\$0.35 /\$1.05
Claude-3-Haiku	API	200K	4K	\$0.25 /\$ 1.25
LLaMA-3-8B	Open Source	8K	-	-
LLaMA-3-70B	Open Source	8K	-	-
Qwen2-7B	Open Source	128K	-	-
Qwen2-57B	Open Source	64K	-	-
Qwen2-72B	Open Source	128K	-	-
ChatGLM4-9B	Open Source	128K	-	-
DeepSeek-v2	Open Source	128K	-	-

Table 1: Comparison of context lengths for various LLMs.

4.1 Models and Inference setup

We evaluated multiple LLMs using LongGenBench, categorizing them into API accessed models and open-source models. For API accessed models, we selected GPT-3.5-Turbo (Ouyang et al., 2022; Brown et al., 2020), GPT-4o (OpenAI, 2024), Gemini-1.5-Flash (Reid et al., 2024), and Claude-3-Haiku (Anthropic, 2024). For open-source models, our selection included LLaMA-3-8B-Instruct, LLaMA-3-70B-Instruct (Meta, 2024), Qwen2-7B-Instruct, Qwen2-57B-A14B-Instruct, Qwen2-72B-Instruct (Bai et al., 2023a), ChatGLM4-9B-Chat (Zeng et al., 2022; Du et al., 2022), and DeepSeek-v2-Chat (DeepSeek-AI, 2024). The API accessed models are configured with a specific maximum output length, which constrains the number of output tokens due to the computational resources and commercial policies of each API provider. Table 1 provides detailed statistics for each model. For open-source models, we remove the Instruct or Chat suffix. The prompt settings and datasets follow the guidelines from the Chain-of-Thought (Wei et al., 2022; Wang et al., 2022; Diao et al., 2024; Fu et al., 2023b; Pan et al., 2024), and API model access is provided through the official website. We assessed all open-source models using the vLLM framework (Kwon et al., 2023), which offers efficient KV cache memory management and a Flash attention (Dao et al., 2022; Dao, 2024) backend. All open source models run on RTX4090 and RTX A6000 servers.

4.2 Task configurations

LongGenBench generates results for three datasets, designated as *LongGenBench-MMLU*, *LongGenBench-GSM8K*, and *LongGenBench-CSQA*. Table 2 details the configurations for the LongGenBench experiments. In this context, K represents the number of questions that the LLM must answer in a single response, while T denotes the number of iterations, also known as query times. The total number of questions addressed is calculated using the formula $K \times T$. To better compare the long-context generation capabilities between API accessed models and open source models, the maximum output length is uniformly set at 4096 tokens in the main experiments. For *LongGenBench-MMLU*, the T value is considered based on the number of categories. Categories with excessively long input prompts are excluded. In our main experiment, we arrange the questions in ascending order based on their length, setting the order within a single query from the shortest to the longest length. A detailed ablation study of this variant is discussed in Section 6.

5 Result

5.1 API Accessed Models

Table 3 displays the performance of various models on the GSM8K and MMLU datasets under two scenarios: Baseline and LongGenBench. The Delta column shows the change in performance when applying LongGenBench relative to the Baseline, with negative values indicated by a downward triangle symbol (∇) signifying performance degra-

Model	LongGenBench					
	GSM8K		MMLU		CSQA	
	K	T	K	T	K	T
GPT-3.5-Turbo	35	20	40	55	80	20
GPT-4o	35	20	40	55	80	20
Gemini-1.5-Flash	35	20	40	55	80	20
Claude-3-Haiku	30	20	40	55	80	20
LLaMA-3-8B-Instruct	30	20	30	52	40	20
LLaMA-3-70B-Instruct	30	20	30	52	40	20
Qwen2-7B-Instruct	30	20	30	52	40	20
Qwen2-54B-A14B-Instruct	30	20	30	52	40	20
Qwen2-72B-Instruct	30	20	30	52	40	20
ChatGLM4-9B-Chat	30	20	30	52	40	20
DeepSeek-v2-Chat	30	20	30	52	40	20

Table 2: Configuration details for the LongGenBench experiment. The table shows the number of questions in one query (K) and the number of iteration times (T).

dation. The results demonstrate that all models undergo a performance degradation when evaluated under the LongGenBench conditions. Notably, GPT-3.5-Turbo and Claude-3-Haiku exhibit the largest Delta on both *LongGenBench-MMLU* and *LongGenBench-GSM8K*, indicating significant challenges in managing long-context generation. Conversely, the Gemini-1.5-Flash model exhibits the smallest performance degradation, suggesting greater robustness and enhanced consistency in handling long-context scenarios.

Model	GSM8K (%)		
	Baseline ↑	LongGenBench↑	Delta∆
GPT-3.5-Turbo	75.1	55.3	-19.8∇
GPT-4o	91.1	75.6	-15.5∇
Gemini-1.5-Flash	86.2	85.0	-1.2∇
Claude-3-Haiku	76.6	55.3	-21.3∇

(a) Performance on GSM8K dataset

Model	MMLU (%)		
	Baseline↑	LongGenBench↑	Delta ∆
GPT-3.5-Turbo	70.0	56.3	-13.7∇
GPT-4o	88.7	79.7	-9.0∇
Gemini-1.5-Flash	79.0	74.7	-4.3∇
Claude-3-Haiku	75.0	48.6	-26.4∇

(b) Performance on MMLU dataset

Table 3: Comparison of baseline and LongGen performance on GSM8K and MMLU datasets with API accessed models.

Figure 3 shows the accuracy distribution of API accessed models in *LongGenBench-GSM8K*. The x-axis represents the question index within a single long-text response, with the maximum index being K. The y-axis indicates the accuracy of the model's responses to these questions. The analysis reveals how the accuracy varies across different

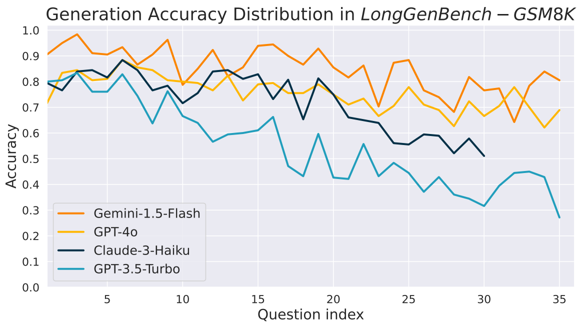

Figure 3: Generation accuracy distribution of API accessed models in *LongGenBench-GSM8K*.

questions for models like GPT-3.5-Turbo, GPT-4o, Gemini-1.5-Flash, and Claude-3-Haiku when they are required to generate answers for K questions simultaneously. The results indicate that all models experience a decline in accuracy as the question index increases. Notably, GPT-3.5-Turbo and Claude-3-Haiku show a more significant decline, suggesting that these models struggle more with maintaining high accuracy over longer sequences of questions. In contrast, Gemini-1.5-Flash maintains relatively higher accuracy, indicating better robustness in handling long-text generation tasks.

5.2 Open Source Models

The results presented in Table 4 provide a comparative analysis of the performance of various open-source models on the GSM8K and MMLU datasets under baseline conditions and using the LongGenBench approach. Several key observations can be made from these results:

Correlation Between Baseline and LongGenBench: There appears to be a general correlation between the baseline performance and the degree of performance degradation observed with the LongGenBench approach. Models with higher baseline performance tend to exhibit smaller performance drops. For example, Qwen2-72B-Instruct and DeepSeek-v2-Chat models, which have high baseline scores, show relatively small Delta values across both datasets. However, there are exceptions, such as LLaMA-3-70B-Instruct, which despite its high baseline performance, exhibits a significant performance drop on both datasets. Additionally, LLaMA-3-8B-Instruct, Qwen2-57B-Instruct, and ChatGLM4-9B-Chat models have the same baseline score on GSM8K, yet their Delta values differ substantially (47.1%, 8.4%, and 10.8%, respec-

Model	GSM8K (%)		
	Baseline↑	LongGenBench↑	Delta∆
LLaMA-3-8B-Instruct	79.6	32.5	-47.1▽
LLaMA-3-70B-Instruct	93.0	83.2	-9.8▽
Qwen2-7B-Instruct	82.3	63.9	-18.4▽
Qwen2-57B-A14B-Instruct	79.6	71.2	-8.4▽
Qwen2-72B-Instruct	91.1	85.7	-5.4▽
ChatGLM4-9B-Chat	79.6	68.8	-10.8▽
DeepSeek-v2-Chat	92.2	86.5	-5.7▽

(a) Performance on GSM8K dataset

Model	MMLU (%)		
	Baseline↑	LongGenBench↑	Delta ∆
LLaMA-3-8B-Instruct	68.4	50.4	-18.0▽
LLaMA-3-70B-Instruct	82.0	71.2	-10.8▽
Qwen2-7B-Instruct	70.5	59.4	-11.1▽
Qwen2-57B-A14B-Instruct	75.4	66.7	-8.7▽
Qwen2-72B-Instruct	82.3	75.8	-6.5▽
ChatGLM4-9B-Chat	72.4	63.0	-9.4▽
DeepSeek-v2-Chat	77.8	72.0	-5.8▽

(b) Performance on MMLU dataset

Table 4: Comparison of baseline and LongGen performance on GSM8K and MMLU datasets with open source models.

tively).

Impact of Model Size on Performance Degradation: Observing models within the same series but with different sizes, such as the LLaMA-3 series and the Qwen2 series, reveals a trend where larger models generally exhibit smaller Delta values. This suggests that increasing model size can mitigate performance degradation in long-context generation tasks. For instance, within the LLaMA-3 series, LLaMA-3-70B-Instruct shows a much smaller Delta compared to LLaMA-3-8B-Instruct across both datasets.

Variation Among Different Model Architectures: Different model architectures demonstrate varying trends in performance degradation. For models within the $7 \sim 9B$ parameter range, such as LLaMA-3-8B-Instruct, Qwen2-7B-Instruct, and ChatGLM4-9B-Chat, there are notable differences in Delta values despite similar baseline performances. For example, LLaMA-3-8B-Instruct has a Delta of 47.1% on GSM8K, while ChatGLM4-9B-Chat has a Delta of only 10.8%, indicating significant variation in how different architectures handle long-context generation tasks.

Consistency Across Tasks for Individual Models: Individual models exhibit consistent trends in performance degradation across different LongGenBench tasks. For instance, LLaMA-3-8B-Instruct consistently shows the largest Delta values on both datasets, indicating a significant drop in performance when generating long-context responses. Conversely, Qwen2-72B-Instruct and DeepSeek-v2-Chat consistently show minimal Delta values, suggesting better resilience in long-context tasks.

These findings underscore the importance of considering both model architecture and size when evaluating the performance of LLMs in long-context generation tasks. The LongGenBench approach effectively highlights the varying capabilities of different models to maintain accuracy over extended text generation, providing valuable insights for further model development and optimization.

Figure 4 illustrates the accuracy distribution of open-source models in *LongGenBench-GSM8K*. The results indicate that all models exhibit a decline in accuracy as the question index increases. Notably, LLaMA-3-8B-Instruct experiences more significant performance degradation, suggesting that this model struggles more with maintaining high accuracy in long-context generation tasks.

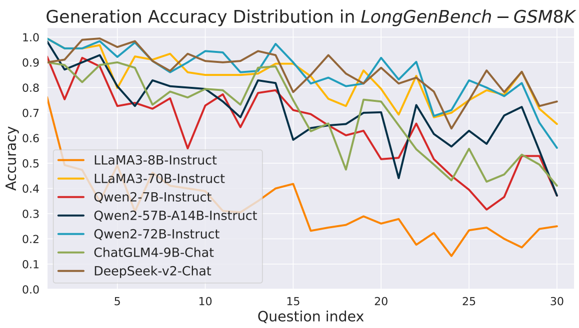

Figure 4: Generation accuracy distribution of open source models in *LongGenBench-GSM8K*.

5.3 Length Distribution

Figure 5 illustrates the output length distribution for various models in the *LongGenBench-GSM8K* task, with experimental configurations as detailed in Table 2. Most models produce output lengths close to or exceeding 3500 characters, although none exceed the 4096-character limit. This data demonstrates that LongGenBench effectively facilitates long-context generation in LLMs.

6 Ablation Studies

Hyperparameters of LongGenBench To gain deeper insights into LongGenBench, we conducted an ablation study focusing on two critical hyperparameters: reconstructive processing and the or-

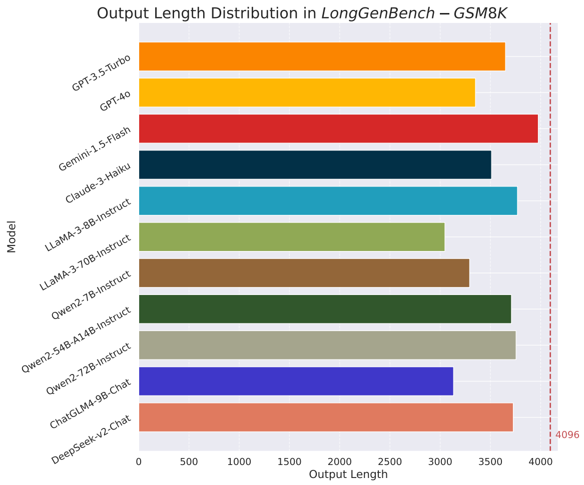

Figure 5: Output Length Distribution in *LongGenBench-GSM8k*.

dering of K questions within a single query. For reconstructive processing, we compared the baseline format with the LongGenBench format. The baseline format follows the CoT (Wei et al., 2022) setting, featuring eight question-answer pairs in sequence. In contrast, the LongGenBench format presents eight questions in advance followed by the corresponding eight answers, with the K questions being addressed subsequently. Regarding the order of K questions, we evaluated three sequences: the original order from the dataset, ascending order from shortest to longest, and descending order from longest to shortest. This ablation study was performed using GPT-3.5-Turbo and Gemini-1.5-Flash on the GSM8k dataset, with the number of iterations set at $T = 20$. Table 5 presents the results of this hyperparameter ablation study, demonstrating that both hyperparameters are crucial for LongGenBench to effectively evaluate the long-context generation capabilities of LLMs.

Model	Format	Order	*LongGenBench GSM8K*
GPT-3.5-Turbo	Baseline	Ascending	53.2
	LongGenBench	Ascending	**55.3**
	LongGenBench	Descending	51.3
	LongGenBench	Normal	47.3
Gemini-1.5-Flash	Baseline	Ascending	82.8
	LongGenBench	Ascending	**85.0**
	LongGenBench	Descending	84.3
	LongGenBench	Normal	**85.0**

Table 5: Performance comparison of different hyperparameter settings.

Evaluating Long Input Comprehension To address the potential concern that performance degradation in LongGenBench may be due to the model's inability to comprehend long inputs rather than its ability to generate long outputs, we conducted additional experiments. Specifically, we designed a set of experiments where the model is provided with a long input containing multiple questions but is required to answer only one specified question at a time. This "long input + short output" setting helps isolate the model's comprehension ability from its generation capacity.

For this experiment, we again used GPT-3.5-Turbo and Gemini-1.5-Flash on the GSM8k dataset. We provided the models with a long input sequence of K questions but instructed them to respond to only one randomly selected question per query. This setup was repeated for $T = 20$ iterations to ensure robust evaluation.The results, shown in Table 6, indicate that both models maintain high accuracy when required to produce short outputs from long inputs. This supports the hypothesis that the primary challenge in LongGenBench lies in the generation of long outputs rather than comprehension of long inputs.

Model	Long Input + Short Output	Long Input + Long Output	Performance Drop
GPT-3.5-Turbo	74.3	55.3	-19.0
Gemini-1.5-Flash	86.1	85.0	-1.1

Table 6: Comparison of model performance in long input + short output versus long input + long output settings.

7 Conclusion

In this study, we introduced LongGenBench, an effective framework designed to evaluate the long-context generation capabilities of language models (LLMs) across multiple datasets. Our experiments included both API accessed and open source models, offering a comprehensive comparison of their performance in long-context generation tasks. The results indicate a correlation between baseline performance and LongGenBench performance, with higher baseline models generally exhibiting smaller declines. Additionally, model size and architecture significantly influence resilience, with larger models and specific architectures demonstrating greater robustness and consistent trends across different LongGenBench tasks. These findings highlight the importance of considering both model architecture and size when evaluating LLMs in long-context generation tasks. The LongGenBench framework effectively showcases the varying capabilities of different models, providing valuable insights for further model development and optimization.

8 Limitations

Our study has several limitations. Firstly, the experiments were conducted on a limited set of models and datasets, which may not fully represent the diversity of available LLMs and tasks. Secondly, we did not explore experiments with larger K values due to constraints on the maximum output tokens imposed by API accessed models. Lastly, we did not include experiments with long-context techniques, which may help mitigate the observed performance degradation. These limitations suggest that further research is needed to generalize our findings across a broader range of models and more extended context scenarios.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 62272122, the Guangzhou Municipal Joint Funding Project with Universities and Enterprises under Grant No. 2024A03J0616, the Hong Kong RIF grant under Grant No. R6021-20, and Hong Kong CRF grants under Grant No. C2004-21G and C7004-22G, the Guangdong Provincial Department of Education Project (Grant No.2024KQNCX028), the Scientific Research Projects for the Higher-educational Institutions (Grant No.2024312096), Education Bureau of Guangzhou Municipality, the Guangzhou-HKUST(GZ) Joint Funding Program (Grant No.SL2024A03J01201), Education Bureau of Guangzhou Municipality, the China Association for Science and Technology (Grant No.XMSB20240711064).

References

AI21. 2024. Introducing jamba: Ai21's groundbreaking ssm-transformer model.

Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *ArXiv preprint*, abs/2307.11088.

Anthropic. 2024. Introducing the next generation of claude.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023a. Qwen technical report. *ArXiv preprint*, abs/2309.16609.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023b. Longbench: A bilingual, multitask benchmark for long context understanding. *ArXiv preprint*, abs/2308.14508.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aydar Bulatov, Yuri Kuratov, Yermek Kapushev, and Mikhail S Burtsev. 2023. Scaling transformer to 1m tokens and beyond with rmt. *ArXiv preprint*, abs/2304.11062.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *ArXiv preprint*, abs/2306.15595.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023b. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *ArXiv preprint*, abs/2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168.

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

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *Preprint*, arXiv:2405.21060.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2024. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1330–1350, Bangkok, Thailand. Association for Computational Linguistics.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *ArXiv preprint*, abs/2307.02486.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *ArXiv preprint*, abs/2402.13753.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. 2024. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *ICML*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, Dublin, Ireland. Association for Computational Linguistics.

Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. 2022. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*.

Daniel Y Fu, Elliot L Epstein, Eric Nguyen, Armin W Thomas, Michael Zhang, Tri Dao, Atri Rudra, and Christopher Ré. 2023a. Simple hardware-efficient long convolutions for sequence modeling. In *International Conference on Machine Learning*, pages 10373–10391. PMLR.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023b. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *Preprint*, arXiv:2305.17306.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *ArXiv preprint*, abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *ArXiv preprint*, abs/2308.16137.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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

Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S. Yu, and Zhijiang Guo. 2024a. Towards understanding factual knowledge of large language models. In *The Twelfth International Conference on Learning Representations*.

Xuming Hu, Xiaochuan Li, Junzhe Chen, Yinghui Li, Yangning Li, Xiaoguang Li, Yasheng Wang, Qun Liu, Lijie Wen, Philip Yu, and Zhijiang Guo. 2024b. Evaluating robustness of generative search engine on adversarial factoid questions. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10650–10671, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Yunpeng Huang, Jingwei Xu, Junyu Lai, Zixu Jiang, Taolue Chen, Zenan Li, Yuan Yao, Xiaoxing Ma, Lijuan Yang, Hao Chen, Shupeng Li, and Penghao Zhao. 2023. Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey. *ArXiv preprint*, abs/2311.12351.

Sam Ade Jacobs et al. 2023. DeepSpeed Ulysses: System optimizations for enabling training of extreme long sequence Transformer models. *ArXiv preprint*, abs/2309.14509.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *ArXiv preprint*, abs/2310.06839.

Gregory Kamradt. 2023. Needle In A Haystack - pressure testing LLMs. *Github*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Dacheng Li, Rulin Shao, et al. 2023. How long can open-source LLMs truly promise on context length?

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2024a. World model on million-length video and language with ringattention. *ArXiv preprint*, abs/2402.08268.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Jiaheng Liu, Zhiqi Bai, Yuanxing Zhang, Chenchen Zhang, Yu Zhang, Ge Zhang, Jiakai Wang, Haoran Que, Yukang Chen, Wenbo Su, et al. 2024b. E^ 2-llm: Efficient and extreme length extension of large language models. *ArXiv preprint*, abs/2401.06951.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024c. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Pedro Henrique Martins, Zita Marinho, and Andre Martins. 2022. ∞-former: Infinite memory transformer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5468–5485, Dublin, Ireland. Association for Computational Linguistics.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. `https://ai.meta.com/blog/meta-llama-3/`. Accessed: 2024-06-07.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *ArXiv preprint*, abs/2305.16300.

OpenAI. 2024. Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`. Accessed: 2024-06-07.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Rui Pan, Shuo Xing, Shizhe Diao, Wenhe Sun, Xiang Liu, KaShun Shum, Jipeng Zhang, Renjie Pi, and Tong Zhang. 2024. Plum: Prompt learning using metaheuristics. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2177–2197, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. 2023. Rwkv: Reinventing rnns for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv preprint*, abs/2403.05530.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7977–7989.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2023. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14590–14604.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhenheng Tang, Yuxin Wang, Xin He, Longteng Zhang, Xinglin Pan, Qiang Wang, Rongfei Zeng, Kaiyong Zhao, Shaohuai Shi, Bingsheng He, et al. 2023. Fusionai: Decentralized training and deploying llms with massive consumer-level gpus. *ArXiv preprint*, abs/2309.01172.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Neil Houlsby, and Donald Metzler. 2022. Unifying language learning paradigms. *ArXiv preprint*, abs/2205.05131.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. 2022. Memformer: A memory-augmented transformer for sequence modeling. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 308–318, Online only. Association for Computational Linguistics.

X.AI. 2024. Announcing grok-1.5.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2023. Effective long-context scaling of foundation models. *ArXiv preprint*, abs/2309.16039.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets long context large language models. In *The Twelfth International Conference on Learning Representations*.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *ArXiv preprint*, abs/2403.04652.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *ArXiv preprint*, abs/2210.02414.

Caiqi Zhang, Zhijiang Guo, and Andreas Vlachos. 2024a. Do we need language-specific fact-checking models? the case of chinese. *Preprint*, arXiv:2401.15498.

Longteng Zhang, Xiang Liu, Zeyu Li, Xinglin Pan, Peijie Dong, Ruibo Fan, Rui Guo, Xin Wang, Qiong Luo, Shaohuai Shi, and Xiaowen Chu. 2023. Dissecting the runtime performance of the training, fine-tuning, and inference of large language models. *Preprint*, arXiv:2311.03687.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024b. Soaring from 4k to 400k: Extending llm's context with activation beacon. *ArXiv preprint*, abs/2401.03462.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, et al. 2024c. ∞-bench: Extending long context evaluation beyond 100k tokens. *ArXiv preprint*, abs/2402.13718.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2023. Pose: Efficient context window extension of llms via positional skip-wise training. In *The Twelfth International Conference on Learning Representations*.

A Experiment Setup and Hyperparameters

A.1 Baseline Setting

The baseline scores referenced in section 5 are derived from official reports to ensure the use of the highest available baseline scores.

A.2 Dataset

The statistics of the datasets used in our study are reported in Table 7.

DATASET	# TRAIN	# TEST
GSM8K (COBBE ET AL., 2021)	7,473	1,319
MMLU (HENDRYCKS ET AL., 2021)	-	14,079
CSQA* (TALMOR ET AL., 2019)	9,741	1,221

Table 7: The statistics of datasets. # TRAIN and # TEST denote the number of training and test samples respectively. *: CSQA do not have publicly available test set labels, so we simply follow the setting by (Wei et al., 2022) to evaluate the performance of the development set.

Table 8 below compares the number of instances in LongGenBench with other long-text benchmarks, demonstrating that LongGenBench has a comparable data size in the long-context benchmark field.

DATASET	# TEST
LONGGENBENCH	$16K$
LONGBENCH (BAI ET AL., 2023B)	$5K$
∞-BENCH (ZHANG ET AL., 2024C)	$5K$
ZEROSCROLLS (SHAHAM ET AL., 2023)	$4K$
L-EVAL (AN ET AL., 2023)	$2K$

Table 8: Comparison of the number of test instances in LongGenBench with other long-text benchmarks, demonstrating that LongGenBench has a comparable data size in the long-context benchmark field.

A.3 LongGenBench Format

Table 12 compares the baseline method with the LongGenBench approach in terms of chat templates used for model interactions. In the baseline method, the system prompt is followed by a series of chain-of-thought (CoT) questions and answers, ending with a real question. In contrast, the LongGenBench approach involves concatenating multiple Chain of Thought (CoT) questions and answers, followed by several real questions, to prompt the model for long-context responses. This method helps evaluate the model's ability to generate coherent and accurate long-context answers across a series of related questions. Additionally, it is easily adaptable to existing benchmarks, allowing for more comprehensive assessments of long-context generation capabilities.

Table 9 presents the system prompt for LongGenBench, which is designed to be straightforward, guiding the LLM to answer each question sequentially.

A.4 API Models

In our experiments, we utilized specific versions of various API accessed models to evaluate their performance on LongGenBench. Table 10 provides the details of the models and their respective versions used in our study.

These versions were selected based on their availability and state-of-the-art performance at the time of experimentation. Each model was tested using the LongGenBench framework to assess their capabilities in handling long-context generation tasks. The results presented in this paper reflect the performance of these specific versions, providing a comprehensive comparison across different models and their configurations.

B Additional Experiments

B.1 API Accessed Models

Figure 6 and Table 11 present the generation accuracy distribution for API accessed models in *LongGenBench-MMLU* and *LongGenBench-CSQA*. The x-axis represents the question index within a single long-text response, with the maximum index being K. The y-axis indicates the accuracy of the model's responses to these questions. The analysis demonstrates how the accuracy varies across different questions for models like GPT-3.5-Turbo, GPT-4o, Gemini-1.5-Flash, and Claude-3-Haiku when they are required to generate answers for K questions simultaneously. The results reveal that all models experience a decline in accuracy as the question index increases, with GPT-3.5-Turbo and Claude-3-Haiku showing more significant declines. Conversely, Gemini-1.5-Flash and GPT-4o maintains relatively higher accuracy, indicating better robustness in handling long-text generation tasks. Since these models do not provide official results for CSQA, we use our replicated baseline score.

LongGenBench System Prompt Exemplars
Answer each question step by step, adhering to the format shown in the examples provided. Start each response with 'Answer_' and introduce the final response with 'The answer is'. Do not repeat the question. Ensure that you respond to all the questions presented, regardless of their number.

Table 9: LongGenBench System Prompt Exemplars

Model	Version
GPT-3.5-Turbo	GPT-3.5-Turbo-0125
GPT-4o	GPT-4o-2024-05-13
Gemini-1.5-Flash	Gemini-1.5-Flash-Preview-0514
Claude-3-Haiku	Claude-3-Haiku-20240307

Table 10: Specific versions of API models used in the experiments.

Model	CSQA (%)		
	Baseline↑	LongGenBench↑	Delta Δ
GPT-3.5-Turbo	75.57	61.88	-13.87▽
GPT-4o	85.75	77.88	-7.87▽
Gemini-1.5-Flash	83.87	82.25	-1.62▽
Claude-3-Haiku	66.75	55.25	-11.50▽

Table 11: Comparison of baseline and LongGen performance on CSQA datasets with API models.

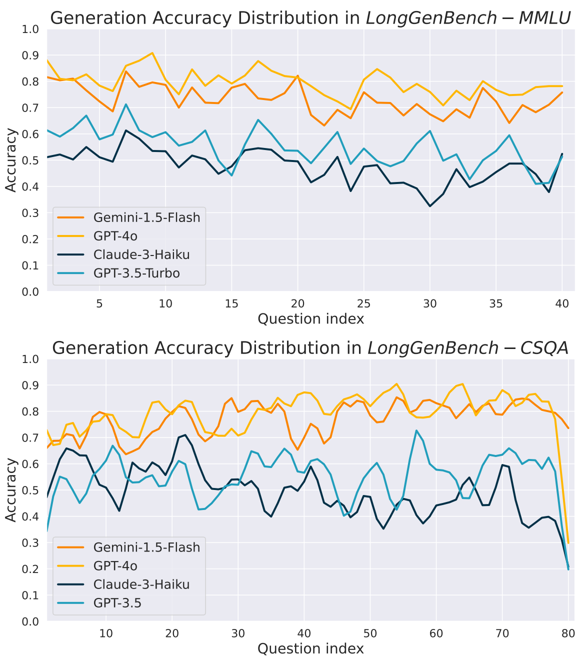

Figure 6: Generation accuracy distribution of API accessed models in *LongGenBench-MMLU* and *LongGenBench-CSQA*.

In addition to the main experiments, we conducted an extended evaluation using the Gemini-1.5-Flash model, which supports a longer maximum output length $8K$ tokens. For this extended evaluation, we set the value of K in the range $40, 50, 60, 70, 80, 90$ and fixed the number of iterations T at 10. This allowed us to investigate the impact of larger K values on model performance in the *LongGenBench-GSM8K* experiment. Figure 7 shows the performance scores as K increases from 40 to 90, with the number of iterations T set to 10. A red dashed line indicates the baseline score of 86.2 for comparison.

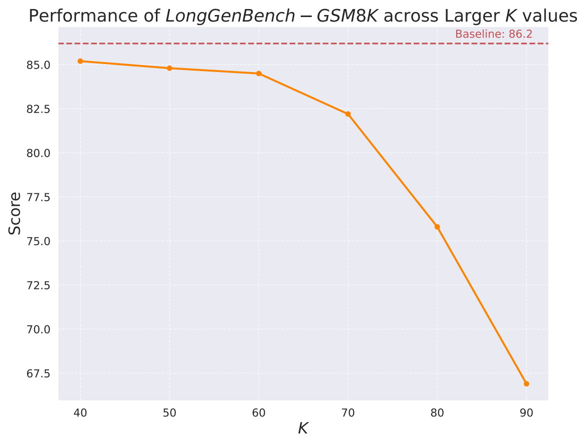

Figure 7: Performance of Gemini-1.5-Flash in *LongGenBench-GSM8K* with larger K values.

B.2 Open Source Model

Figure 8 and Table 13 illustrates the generation accuracy distribution for open source models in *LongGenBench-MMLU* and *LongGenBench-CSQA*. Similar to the API accessed models, the x-axis represents the question index, and the y-axis represents the accuracy of responses. The results indicate that all open source models exhibit a decline in accuracy as the question index increases. Notably, LLaMA-3-8B-Instruct experiences significant performance degradation, suggesting that this model struggles with maintaining high accuracy in long-context generation tasks. In contrast, larger models such as LLaMA-3-70B-Instruct and Qwen2-72B-Instruct demonstrate greater resilience, maintaining higher accuracy across longer sequences of questions. Since these models do

Method	Template
Baseline	{System Prompt} {CoT Question_1}{CoT Asnwer_1} . . . {CoT Question_8}{CoT Asnwer_8} {Real Question}
LongGenBench	{System Prompt} {CoT Question_1}· · ·{CoT Question_8} {CoT Asnwer_1}· · ·{CoT Asnwer_8} {Real Question_1} · · · {Real Question_K}

Table 12: Model chat templates.

not provide official results for CSQA, we use our replicated baseline score.

MODEL	CSQA (%)		
	BASELINE↑	LONGGENBENCH↑	DELTA Δ
LLAMA-3-8B-INSTRUCT	73.70	69.50	-4.20∇
LLAMA-3-70B-INSTRUCT	81.82	80.13	-1.69∇
QWEN2-7B-INSTRUCT	78.13	77.25	-0.88∇
QWEN2-57B-A14B-INSTRUCT	80.26	80.75	+0.49Δ
QWEN2-72B-INSTRUCT	87.75	85.50	-2.25∇
CHATGLM4-9B-CHAT	85.42	82.37	-3.05∇
DEEPSEEK-V2-CHAT	84.77	82.87	-1.9∇

Table 13: Comparison of baseline and LongGen performance on CSQA datasets with open source models.

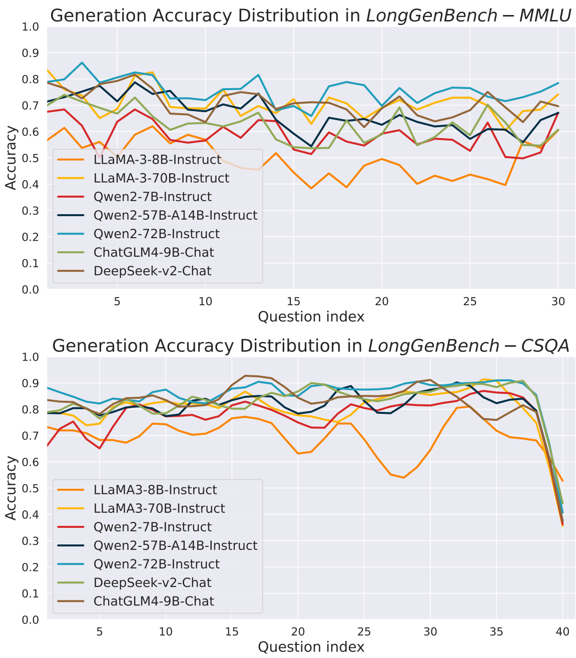

Figure 8: Generation accuracy distribution of open source models in *LongGenBench-MMLU* and *LongGenBench-CSQA*.

B.3 Cost

In this section, we compare the costs between the Needle-In-A-Haystack (NIAH) test and LongGenBench. For NIAH, costs are calculated for a 128K token test. For LongGenBench, calculations are based on the same K and T values used in the GPT-4o configuration, as specified in Table 2, and applied across the datasets *LongGenBench-GSM8K*, *LongGenBench-MMLU*, and *LongGenBench-CSQA*. The pricing is based on the GPT-4o rates, which are $5 per million input tokens and $15 per million output tokens.

As shown in Table 14, the NIAH test incurs a significantly higher cost compared to LongGenBench. Specifically, the NIAH test requires 8192K input tokens and 0.1K output tokens, resulting in a total cost of $40. In contrast, LongGenBench requires 300K input tokens and 340K output tokens, resulting in a much lower total cost of $6.6.

Method	Input Tokens	Output Tokens	Price
NIAH	8192K	0.1K	$40
LongGenBench	300K	340K	$6.6

Table 14: Comparison of Costs Between Needle-In-A-Haystack (NIAH) and LongGenBench Tests

C Exemplars Generated by LLMs Using LongGenBench

Tables 15 and 16 provide examples of responses generated by LLMs using LongGenBench, where each model generated K answers in a single long-context response. These examples demonstrate that the models can remember the content and sequence of the K questions and maintain consistency in logical flow during generation.

D Licenses

For the evaluation dataset, all the datasets, including GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021), CSQA (Talmor et al., 2019) are released under MIT license.

Exemplars

Answer_1:
Terry eats 2 yogurts a day, so in 30 days, he eats $30 * 2 = 60$ yogurts. Since yogurts are on sale at 4 for \$5.00, he spends $60/4 * \$5.00 = \$75.00$ on yogurt over 30 days. ❌

Answer_2:
There are 25 oranges in total, and 20% of them are unripe, so 20% of 25 is $0.20 * 25 = 5$. 1 is bad and 2 are sour, so from $25 - (1 + 2 + 5) = 25 - 8 = 17$ Therefore, 17 oranges are good. ✅

Answer_3:
There are 6 clusters with $6 * 20 = 120$ raspberries. Together with the 67 individual fruit, there are $120 + 67 = 187$ raspberries in total. ✅

Answer_4:
Lloyd produces 252 eggs per day, so in a week, he produces 252 * 7 = 1764 eggs. He sells them for \$2 per dozen, so he sells $1764/12 = 147$ dozens per week. Therefore, his weekly income from eggs is $147 * \$2 = \$294$. ✅

Answer_5:
If there are twice as many boys as girls in the school and there are 60 girls, then there are $2 * 60 = 120$ boys. With 5 students to every teacher, there are $60 + 120 = 180$ students. Therefore, there are 180 / 5 = 36 teachers. ✅

......

Answer_24:
Each school sent 2 teams with 5 players and 2 coaches, so each school sent $2 * (5 + 2) = 14$ people. With 4 schools, there are therefore $4 * 14 = 56$ people in total. ❌

Answer_25:
If Dana spends 1/3 of 6 hours running, she runs $1/3 * 6 = 2$ hours. She travels at a rate of $3 * 2 = 6$ miles per hour. If she spends 2/3 of 6 hours walking, she walks $2/3 * 6 = 4$ hours. She travels at $1/4 * 3 = 0.75$ miles per hour, so she walks $4 * 0.75 = 3$ miles. In total, she travels $2 * 6 + 4 * 3 = 12 + 12 = 24$ miles in 6 hours. ❌

Answer_26:
The final price of Stephen's groceries is $25\% + \$3.00 + \$4.00 = \$13.00$ more than \$40.00, so the final price is $\$40.00 + \$13.00 = \$53.00$. ❌

Answer_27:
In the first half of the year, Aleena pays $\$140 * 6 = \$840$. After the discount, in the second half of the year, she pays $10\% * \$140 * 6 = \$84$ less, so she pays $\$840 - \$84 = \$756$. Thus, the total amount she paid for the streaming service by the end of the year is $\$840 + \$756 = \$1596$. ✅

Answer_28:
The truck and driver weigh 3755 pounds, leaving a maximum capacity of $5000 - 3755 = 1245$ pounds. So up to $1245/15 = 83$ boxes can be loaded onto the truck. ✅

......

Answer_32:
Sophia travels 100 miles on 4 gallons of gas, so she uses $100/4 = 25$ miles per gallon. On one tank of 12 gallons, she will be able to drive $25 * 12 = 300$ miles. ✅

Answer_33:
For repairing 6 truck tires, the revenue is $6 * \$60 = \$360$. For repairing 4 car tires, the revenue is $4 * \$40 = \$160$. The difference in revenue is $\$360 - \$160 = \$200$ more. ❌

Answer_34:
Meredith spent 4 hours on each blog article, so 5 articles took $5 * 4 = 20$ hours on Monday. On Tuesday, she wrote 2/5 more articles, which is $2/5 * 5 = 2$, making a total of $5 + 2 = 7$ articles. Then on Wednesday, she wrote twice the number of articles as on Tuesday, so $7 * 2 = 14$ articles. All in all, Meredith spent $20 + (7 * 4) + (14 * 4) = 20 + 28 + 56 = 104$ hours writing articles in the three days. ✅

Answer_35:
If Polly holds the sapphire gemstone, she can throw the gold javelin three times as far, so she can throw it $400 * 3 = 1200$ feet. Therefore, holding the sapphire gemstone, she can stand $1200 - 1000 = 200$ feet outside the reach of the dragon's flames and still hit the dragon. ✅

Table 15: Exemplars for GPT-3.5-Turbo generated response from *LongGenBench-GSM8K*.

Exemplars

Answer_1:
Let's think step by step. Bubblesort has $O(n^2)$ running time in both worst and average cases. Mergesort has $O(nlogn)$ running time in both worst and average cases. Heapsort has $O(nlogn)$ running time in both worst and average cases. Quicksort has $O(n^2)$ running time in the worst case and $O(nlogn)$ running time on average. The answer is (D). ✅

Answer_2:
Let's think step by step. Insertion sort is heavily dependent on the initial ordering of the input. Quicksort is heavily dependent on the initial ordering of the input. Merge sort is least dependent on the initial ordering of the input, because its performance is consistent regardless of the ordering of the input. Selection sort is also heavily dependent on the initial ordering of the input. The answer is (C).✅

Answer_3:
Let's think step by step. We can think of a K-ary tree as being composed of one nonterminal root node, K nonterminal child nodes of the root, K^2 nonterminal child nodes of each of those nodes, etc. Thus, a K-ary tree of depth N is composed of: $1 + K + K^2 + ... + K^N$ nonterminal nodes and $(K^N - 1)/(K - 1)$ total nodes. As the depth of the tree increases, the proportion of nonterminal nodes relative to the total number of nodes approaches: $(1 + K + K^2 + ... + K^N)/((K^N - 1)/(K - 1)) = ((K^(N + 1) - 1)/(K - 1))/((K^N - 1)/(K - 1)) = (K^(N + 1) - 1)/(K^N - 1) = (K^(N + 1) - 1)/(K^N - 1) * (K/K) = (K^(N + 2) - K)/(K^(N + 1) - K) = (K^(N + 1) - 1)/(K^N - 1) + 1 = (1 + K + K^2 + ... + K^(N + 1))/(1 + K + K^2 + ... + K^N)$ which as N increases approaches: $K/(K - 1)$. The answer is (B). ❌

Answer_4:
Let's think step by step. We can see that {AND, OR} is not complete because we cannot express NOT. We can see that {AND, NOT} and {NOT, OR} are both complete. We can also see that {NAND} is complete because NAND can express all other boolean operations. The answer is (C). ✅

......

Answer_19:
Let's think step by step. The IP protocol is primarily concerned with routing packets through the network. It defines the format of packets and the addressing scheme used to route packets between different networks. The answer is (A). ✅

Answer_20:
Let's think step by step. TCP protocol is responsible for reliable delivery of large (multi-packet) messages between machines that are not necessarily directly connected. It provides a reliable connection-oriented service, ensuring that packets arrive in the correct order and are not lost. The answer is (C). ✅

Answer_21:
Let's think step by step. A common divisor must divide both integers. The set of common divisors of two integers is a subset of the divisors of each integer. Thus, the set {-6,-2, -1, 1, 2, 6} could possibly be the set of all common divisors of two integers. The answer is (A). ❌

......

Answer_38:
Let's think step by step. If the owner is allowed to delete a file, dangling links may result. This is true, because if the owner deletes the file, the link count in the shared file node will be decremented. However, the links to the file in other users' directories will still point to the shared file node. If the owner deletes the file, these links will become dangling links. The answer is (C). ✅

Answer_39:
Let's think step by step. Giving priorities to processes and ordering the wait queues by priority will not deal effectively with the problem of deadlock. This is because deadlock can occur even if processes have priorities and wait queues are ordered by priority. For example, consider a system with two processes, P1 and P2, and two resources, R1 and R2. P1 holds R1 and requests R2, while P2 holds R2 and requests R1. Deadlock will occur even if P1 has a higher priority than P2. The answer is (A). ✅

Answer_40:
Let's think step by step. Indirect representation noticeably increases compilation time. This is false, because indirect representation does not noticeably increase compilation time. Direct representation decreases the time needed to access components of a variable. This is true, because direct representation allows for faster access to components of a variable, as the compiler does not need to dereference a pointer to access the components. When the storage size of some private component of a variable changes, indirect representation minimizes the number of recompilations of source modules that must be performed. This is true, because indirect representation allows for changes to the size of private components without requiring recompilation of source modules that use the variable. The answer is (D). ✅

Table 16: Exemplars for Gemini-1.5-Flash generated response from *LongGenBench-MMLU*.