LLMs Can't Generalize Far on Grade-School Math, But Long Chain-of-Thought Can Help

Anonymous ACL submission

Abstract

We investigate how large language models (LLMs) generalize to math problems requiring extended reasoning[1]. Specifically, we assess LLM performance on GSM8K-like problems, which require an increasing number of operations to solve, and conduct a detailed analysis of the models' responses. Our findings reveal a significant decline in accuracy as problem complexity increases, while thinking time (both token length and reasoning steps) naturally grows. Additionally, we explore whether using more complex Chain-of-Thought (CoT) prompts can enhance generalization and examine the performance of reasoning models (i.e., o1-like models that generate long CoTs). Although reasoning models show better length generalization, this limitation persists, highlighting inherent constraints in current LLMs' ability to learn from shorter, simpler examples and apply that knowledge to longer, more complex tasks.

1 Introduction

Large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023; OpenAI, 2024a; Yang et al., 2024; DeepSeek-AI, 2024) have demonstrated impressive reasoning abilities in domains such as mathematics (Hendrycks et al., 2021) and programming (Jain et al., 2024). However, despite these advancements, LLMs still struggle to generalize their success on simpler problems to more complex ones. For example, while LLMs perform well on arithmetic problems involving the addition or multiplication of smaller numbers, their performance often declines as the number of digits increases (Jelassi et al., 2023). Since more complex problems typically have less training data (Anil et al., 2022), it is crucial for LLMs to learn from shorter examples and generalize this knowledge to longer, more complex tasks.

[1]We will release code and data upon publication.

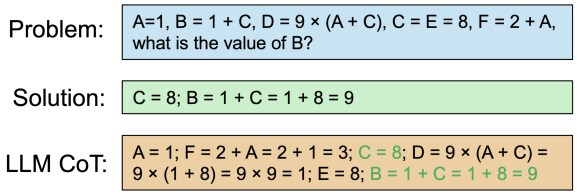

Figure 1: An example of a grade-school math problem, where letters represent the parameters for simplicity. We also present the correct solution with necessary reasoning steps and LLM CoT with necessary steps highlighted in green.

A key technique for enabling reasoning abilities in LLMs is chain-of-thought (CoT) prompting (Nye et al., 2021; Wei et al., 2022), which encourages models to generate intermediate reasoning steps before arriving at a final answer. While CoT has shown promise in facilitating length generalization for simple symbolic reasoning tasks, such as the Coin flip task, it still fails to generalize effectively for more complex reasoning tasks, as detailed in Section 2. However, the potential of CoT for improving length generalization in mathematical reasoning remains an area worthy of further exploration. Previous studies (Fu et al., 2022; Jin et al., 2024) have demonstrated that more complex CoTs can significantly improve mathematical reasoning, suggesting that this approach might also help with length generalization, as models may learn more complex reasoning patterns through such extended CoTs.

Recently, OpenAI's o1 (OpenAI, 2024b) models and its replicas (Qwen Team, 2024; DeepSeek-AI, 2025; Kimi Team, 2025) have made significant breakthroughs in reasoning. A key distinguishing feature of these models is their ability to scale inference compute with long CoTs, incorporating strategies such as recognizing and correcting mistakes, breaking down complex steps, and iterating on alternative approaches. These reasoning models have shown great potential in solving logic grid

puzzles and generalizing to more complex ones (Lin et al., 2025). However, it remains unclear whether they can achieve length generalization in mathematical reasoning tasks.

In this work, we focus on length generalization in LLMs for solving grade-school math problems. Here, "length" refers specifically to reasoning length, which may not necessarily correspond to longer input lengths as seen in previous works (Wei et al., 2022; Anil et al., 2022). While the saturation of the GSM8K (Cobbe et al., 2021) dataset might suggest that LLMs have mastered this type of problem, we emphasize that most GSM8K problems require fewer than five reasoning steps.

Therefore, the ability to extend the complexity of these problems is crucial for our study. Using a synthetic framework (Ye et al., 2025) (see Section 3 for more details), we can generate an infinite number of GSM8K-like problems with controlled complexity—specifically, the number of parameters in the problems and operations required to reach a solution—providing an ideal testbed for our investigation.

Through our evaluation of various LLMs—three conventional models (Qwen2.5-72B-Instruct, Llama3.3-70B, and DeepSeek-V3) and two reasoning models (QwQ-32B-Preview and DeepSeek-R1)—we found that CoT prompting provides limited improvement in length generalization. Performance drops significantly as problem complexity increases, and using more complex CoT prompts yields only marginal gains for simpler problems, while still failing for more complex ones. However, we observed that reasoning models, particularly DeepSeek-R1, show better generalization and perform better as complexity increases.

Instead of solely focusing on the accuracy of solving problems of varying complexity, we also analyze the responses of LLMs from different perspectives, such as how well the responses cover the necessary parameters, the number of reasoning steps taken, and the token length. Through this analysis, we observed several interesting patterns. For example, all models naturally require more compute as problem complexity increases. Additionally, redundant reasoning steps are frequently observed. As shown in Figure 1, only two steps are necessary in the LLM CoT, while the rest are irrelevant. While redundancy tends to decrease for conventional LLMs as problems become more complex, it increases for reasoning models.

Our work makes the following key contributions:

- We explore how well conventional LLMs generalize to GSM8K-like problems of varying complexity using CoT prompting.
- We investigate how well reasoning models generalize with long CoTs.
- We provide a comprehensive analysis of models' responses and insights into the reasoning process when handling problems of varying complexity.

2 Related Work

Wei et al. (2022) demonstrated that chain-of-thought (CoT) prompting can facilitate length generalization for inference-time inputs longer than those seen in the few-shot exemplars, specifically for two symbolic reasoning tasks. The first task, *Last Letter Concatenation*, asks the model to concatenate the last letters of words (e.g., "Byron Kim" $\rightarrow$ "bk"). The second task, *Coin Flip*, requires the model to determine whether a coin remains heads up after certain actions, such as flipping or not flipping the coin (e.g., "A coin is heads up. Craig flips the coin. Alice does not flip the coin. Is the coin still heads up?" $\rightarrow$ "no").

However, their exploration scaled only from two reasoning steps to four steps. Further scaling, conducted by Stechly et al. (2024), revealed that while generalization holds for the Coin Flip task (with accuracy only dipping below 90% at 31-step problems), the performance on Last Letter Concatenation drops rapidly to near zero when the number of words increases to 20. Stechly et al. (2024) also observed a similar decline in performance for two other tasks: Blockworld, where models must stack blocks in the required order, and Multi-step Arithmetic on Single-digit Numbers, which involves simplifying parenthesized expressions with repeated applications of the four basic arithmetic operations on single-digit numbers. Recently, Lin et al. (2025) also found that accuracy declines significantly as problem complexity increases for logic grid puzzles.

In addition to examining how CoT prompting might assist with length generalization, various other approaches have been tested for their potential to help. Several studies (Zhou et al., 2024; Jelassi et al., 2023) have found that appropriate position encoding can also facilitate length generalization for addition of two numbers. Anil et al.

(2022) demonstrated that both standard fine-tuning and scratchpad fine-tuning fail to achieve length generalization for parity (a classical learning problem that requires predicting whether a bit-string has an even or odd number of ones) and variable assignment (involving Python programs where each line contains a boolean variable assignment, and the model is tasked with outputting the value of the last variable by keeping track of previous assignments). Furthermore, Lin et al. (2025) found that while model growth, training data expansion, and increasing the generation sample size yield modest improvements for logic grid puzzles, a backtracking-based approach with expanded reasoning steps (reasoning models with long CoTs) significantly boosts accuracy.

3 Experiment

In this section, we describe the data generation process, the prompt setup, and the analysis methods employed.

3.1 Data Generation

We generated our data using a synthetic generation pipeline designed to create grade-school math problems similar to those in the GSM8K dataset. The pipeline begins by constructing a structure graph, from which instance parameters are selected for the dependency graph. The problems are then formulated based on the dependencies among the parameters. Finally, the solution is represented as a sequence of sentences outlining the necessary steps to solve the given problem. Lastly, we describe the implementation details for generating our dataset.

Structure graph. The process begins by organizing hierarchical categories into layers, with each layer containing various items. For instance, as shown in Figure 2, there are two layers: *District* and *Supermarket*. The *District* layer may include items such as *Manhattan* and *Chaoyang*, while the *Supermarket* layer includes *Costco* and *Walmart*. A structure graph is then constructed for each math problem by connecting items from different layers. For example, linking *Manhattan* to *Walmart* represents "the number of Walmart available in each Manhattan" and generates a new instance parameter, *Manhattan's Walmart*. Abstract parameters, such as "the number of supermarket available in each Chaoyang" are excluded in structure graph.

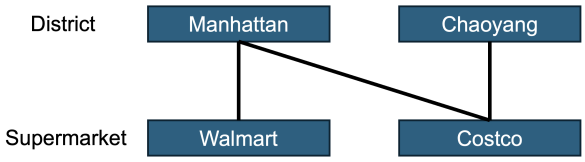

Figure 2: An example of a structure graph. There are two layers: *District* and *Supermarket*, each containing two items. The edges between the items represent parameters; for example, the edge between *Manhattan* and *Walmart* signifies "the number of Walmart in each Manhattan."

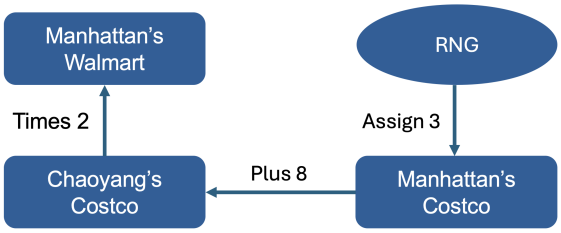

Figure 3: An example of a dependency graph. *RNG* stands for random number generator. The dependencies among the parameters are represented as a directed acyclic graph. For instance, *Manhattan's Costco* plus 8 results in *Chaoyang's Costco*. Implicit dependencies are excluded for simplicity.

Dependency graph. A dependency graph is created to illustrate the relationships among parameters. Each instance parameter depends on up to four other parameters, with one of them potentially being *RNG* (random number generator). For example, as shown in Figure 3, the parameter *Manhattan's Costco* is assigned the value 3, which is randomly generated, while *Manhattan's Walmart* is set to be twice the value of *Chaoyang's Costco*. Worth mentioning, there also exists implicit dependency. For example, *Chaoyang's Supermarket* will be equal to *Chaoyang's Costco*, as *Chaoyang* only has one type of *Supermarket*.

Problem generation. The problem is written in English, with each instance parameter described in a sentence. Abstract parameters are not directly described, but their relationships can be implied through the structure graph. The sentences are shuffled to increase difficulty, and one parameter is selected as the subject of a question and placed at the end. An example of a problem is shown below.

Problem: The number of each Manhattan's Walmart is 2 times more than each Chaoyang's Costco. The number of each Chaoyang's Costco is 8 more than the number of each Manhattan's Costco. The number of each Manhattan's Costco is 3. How many Costco does Chaoyang have?

Solution construction. The solution is presented as a sequence of sentences describing the necessary steps toward solving the given problem — also known as Chain-of-Thought (CoT). For each parameter necessary for answering the final question, a random letter is assigned, and a sentence is used to describe its computation. Throughout this paper, we perform arithmetic operations modulo 10 to avoid errors arising from computations involving large numbers. An example of a solution is shown below.

Solution: Define Manhattan's Costco as A; A = 3. Define Chaoyang's Costco as B; B = 8 + A = 8 + 3 = 1.

Implementation details. For the hierarchical categories, we use four layers: *District*, *Supermarket*, *Product*, and *Ingredient*, each containing 100 items. There are two hyperparameters for difficulty control: ip, representing the number of instance parameters, and op, representing the number of solution operations. We set the $ip \leq 20$ while op will take every even number from 2 to 20. For each value of op, we generate 100 problems. It is worth mentioning that we break computations into binary operations; for example, $a = 1 + 3 + 7$ will take two ops as it will be broken into $a = 1 + b$ and $b = 3 + 7$.

3.2 Prompt Setup

We describe the process of forming the basic prompt here. In certain experiments, CoT examples will be added, as detailed in Section 4. All prompts include domain information, specifying that calculations should be performed modulo 10, and describe the scenario involving the hierarchical categories. They also provide guiding principles on how to handle parameters based on their existence. Additionally, the prompts include formatting requirements for how the solution should be structured: beginning with the definition of a parameter, followed by the computation. For each problem, specific background information is provided, as shown below, along with the problem description outlined in the previous section.

Background: Thera are two types of District: Manhattan and Chaoyang. There are 2 types of Supermarkets: Walmart and Costco. Each Manhattan has Walmart and Costco. Each Chaoyang has Costco.

3.3 Analysis Setup

We focus on four main aspects when analyzing the responses of LLMs.

Accuracy. We measure accuracy for problems requiring each op individually and investigate how well LLMs generalize to problems with longer reasoning lengths. The answers are extracted as the last number from the response.

Parameter Hit. In the correct solution, every parameter is necessary toward calculating the final query parameter. Thus, we measure how close the response is to the correct solution by computing the percentage of necessary parameters that are mentioned in the response. Let $P_{\text{necessary}} = \{p_1, p_2, \ldots, p_n\}$ be the set of necessary parameters. The parameter hit is given by:

$$\text{Parameter Hit (\%)} = \frac{k}{n} \times 100$$

where k is the number of necessary parameters mentioned in the response, and n is the total number of necessary parameters.

To check the existence of the necessary parameters, we begin by segmenting the response into individual sentences. Next, we examine whether two items within a single parameter exist, as the same parameter can be expressed in various ways. For instance, *Manhattan's Walmart* could also be phrased as *Walmart in Manhattan*.

Reasoning step. For LLMs that are able to follow the format requirements — starting by defining a parameter as a random letter and then describing the computation, as mentioned in the previous section — we can easily count the reasoning steps by using regular expressions to count the number of occurrences of "define" in the response. For those that do not follow the format, we have specific methods for counting, which are detailed in Section 4.

Token length. Token length is the most straightforward way to measure the compute LLMs put

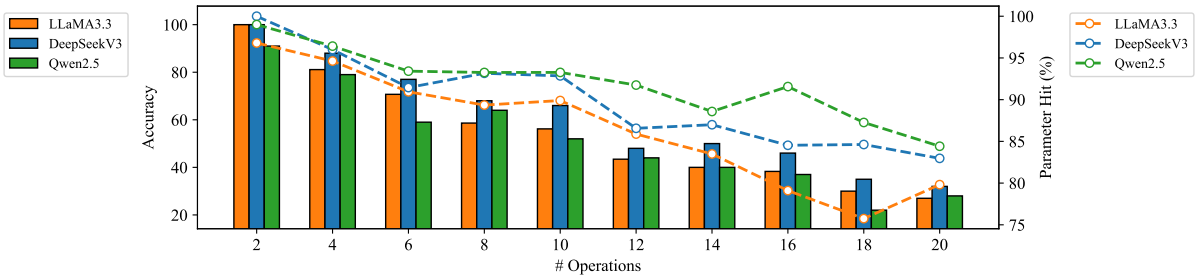

Figure 4: Accuracy and average parameter hit rate of LLMs across different complexity levels of GSM8K-like problems. A pronounced drop in accuracy is observed, while the decline in parameter hit rate is smaller.

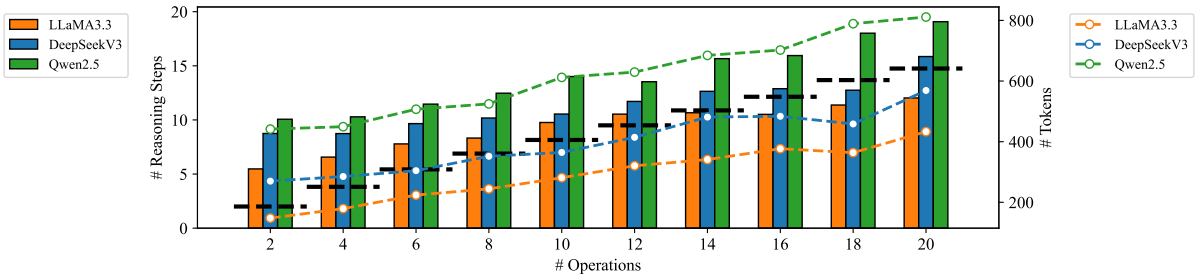

Figure 5: Average number of reasoning steps (with the necessary reasoning steps indicated by the black dashed lines) and tokens for LLMs. Both reasoning steps and tokens increase as the problems become more complex. The plot also shows that more redundant steps are present when the problems are simpler.

into solving the problems. For each value of op, we compute the mean token length.

4 Results

In this section, we present the results, beginning with how three conventional LLMs generalize with Chain-of-Thought (CoT) prompting as the reasoning length of the problems increases. We then investigate how increasingly more complex CoT can help one of the LLMs generalize, and finally, we explore how reasoning models — those that generate long CoTs for the problems — perform in terms of length generalization. Additionally, we provide a comprehensive analysis of the models' responses and share our findings.

4.1 CoT Prompting

Wei et al. (2022) demonstrated that CoT can facilitate length generalization in two symbolic reasoning tasks. However, after scaling these tasks, Stechly et al. (2024) found that such generalization may not hold over longer reasoning lengths. Our experiment confirms that while CoT can aid in length generalization, this generalization does not hold for math reasoning tasks as the complexity increases.

Setup. We use a 5-shot CoT prompting, with each shot including a solution that requires 4 operations. Additionally, we prepend the phrase "let's think step by step" (Kojima et al., 2022) to the beginning of each solution. We test the models Qwen2.5-72B-Instruct, Llama3.3-70B, and DeepSeek-V3. Among the three models, Llama occasionally fails to output a valid response, resulting in fewer responses for some operations (on average, there are 93 responses for each operation). The temperature is set to 0 for all LLMs.

Both Llama3.3-70B and DeepSeek-V3 follow the format well, allowing us to easily count the reasoning steps and parameters using regular expressions. In contrast, Qwen2.5-72B-Instruct occasionally fails to adhere to the format, so we use additional regular expressions to capture those cases and exclude instances where it's difficult to obtain a valid result (on average, 12 instances are excluded per operation).

Results. As shown in Figure 4, LLMs do not generalize well as the complexity of the problems increases. The accuracy decreases from near perfect for $op = 2$ to around 30% for $op = 20$. Meanwhile, LLMs become less able to identify the necessary parameters for a correct solution as the problems grow more complex, which explains the drop in accuracy.

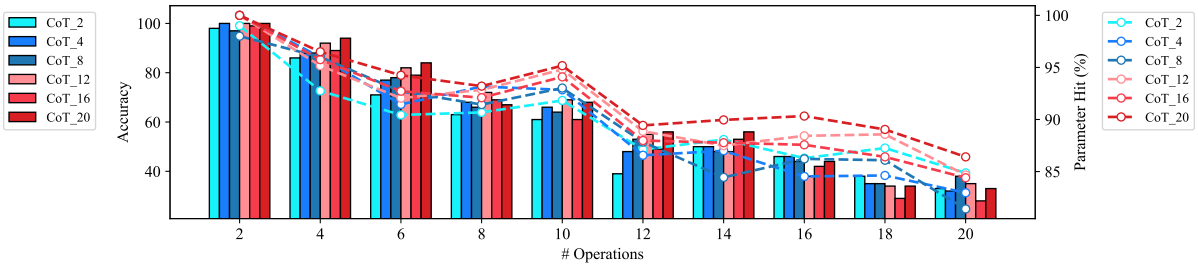

Figure 6: Accuracy and average parameter hit rate of DeepSeek-V3 across CoTs of varying complexity (e.g., "CoT_2" refers to CoT consist of 2 operations). We observe that more complex CoTs lead to performance gains when the problems are not too complex, and consistently result in a higher parameter hit rate.

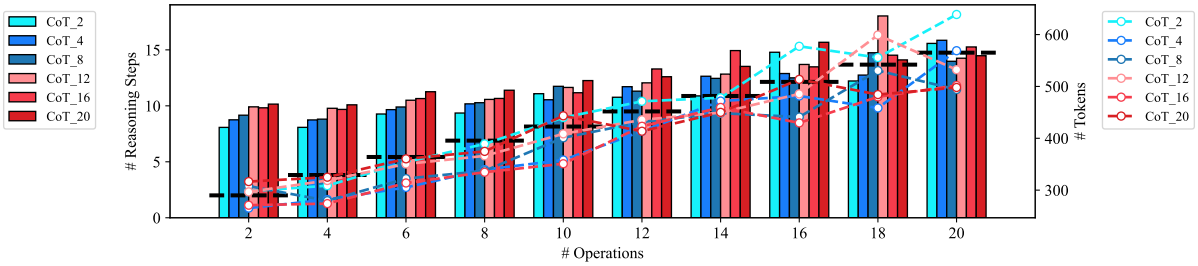

Figure 7: Average number of reasoning steps (with necessary reasoning steps marked by black dashed lines) and token counts for DeepSeek-V3. While more complex CoTs generally increases the number of reasoning steps, it does not always correspond to longer token length. For instance, "CoT_2" results in a higher token count but fewer reasoning steps; an explanation for this discrepancy is provided in Section 4. Additionally, we note an overall reduction in redundancy.

We also observe that simply covering the necessary parameters does not directly lead to higher accuracy. For example, as the number of operations increases, Qwen2.5-72B-Instruct achieves a higher parameter hit rate but exhibits lower accuracy.

Figure 5 presents our further investigation into the CoTs generated by LLMs. We find that LLMs tend to generate more tokens for more difficult problems, indicating that they naturally increase their reasoning time when they find the problem harder to solve. Additionally, the increase in tokens corresponds to a rise in the number of reasoning steps.

However, we observe that many reasoning steps are redundant. For example, for $op = 2$, only 2 steps are necessary, but there are, on average, more than 5 steps. Interestingly, the redundancy decreases as the number of operations increases. In some cases, certain LLMs even take fewer steps than necessary (e.g., Llama3.3-70B and DeepSeek-V3 for $op = 18$), indicating that these models may not be putting enough compute with complex problems. This lack of thorough reasoning can directly lead to a decline in accuracy.

4.2 Increasingly More Complex CoT Prompting

Recent studies (Fu et al., 2022; Jin et al., 2024) have shown that more complex CoTs can lead to significant performance improvements in math word reasoning tasks. We investigate whether increasing the complexity of CoTs can help LLMs generalize to problems that require longer reasoning lengths.

Setup. We create 5-shot CoTs that require an increasing number of operations to solve. We focus our investigation on DeepSeek-V3 in this setting, as it is the most capable model and adheres well to the format requirements. Additionally, we prepend the phrase "let's think step by step" to the beginning of each solution, with the temperature set to 0.

Results. In Figure 6, we observe that more complex CoT prompts lead to some gains for problems with fewer operations, but these gains diminish as the problems increase in complexity. While more complex CoTs do not result in better performance for more complex problems, they do bring the response closer to the correct solution, as the parameter hit rate increases with the complexity of the CoT for most operations.

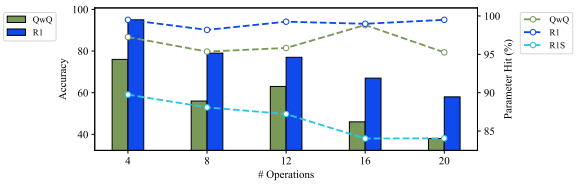

Figure 8: Accuracy and average parameter hit rate of reasoning models. "R1S" refers to the summary part of DeepSeek-R1's response. Since the answers in the summary are always consistent with those in the reasoning content, the accuracy for both parts is identical.

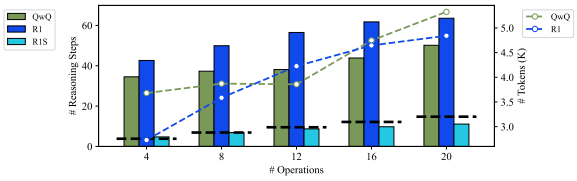

Figure 9: Average number of reasoning steps (with necessary reasoning steps marked by black dashed lines) and token count for reasoning models. For reasoning steps, we separately count the reasoning content and summary of DeepSeek-R1, while the token count is the sum of both, with the summary comprising less than one-tenth of the reasoning content.

In Figure 7, we observe that more complex problems tend to have fewer redundant reasoning steps. Additionally, more complex CoT prompts generally lead to longer reasoning sequences. Interestingly, we find that more complex CoT prompts do not result in more output tokens, whereas the simplest CoT prompts ("CoT_2") tend to generate more tokens, leading in token length for many operations, while with fewer reasoning steps.

Upon manual inspection, we attribute this discrepancy between token length and reasoning steps to the fact that simpler CoT prompts often lead to solutions that directly reference a parameter, rather than using a letter. This often leads to solutions[2] as illustrated below. We hypothesize that this phenomenon arises because models may struggle to learn the topological order effectively with simpler CoTs, as they involve too few reasoning steps. As a result, while the number of reasoning steps remains constant, the token count increases.

> **Solution:** Define Chaoyang's Costco as B; B = 8 + Manhattan's Costco = 8 + 3 = 1. Define Manhattan's Costco as A; A = 3.

4.3 Prompting Reasoning Models

Recent studies (DeepSeek-AI, 2025; Kimi Team, 2025) on reasoning models have observed that models progressively enhance their performance on reasoning tasks through reinforcement learning. Accompanying this improvement is the emergence of notable long Chain-of-Thought (CoT) reasoning, where LLMs demonstrate the ability to backtrack and correct errors. Our experiments aim to explore whether such extended CoTs can facilitate length generalization in mathematical reasoning tasks.

Furthermore, studies (Chen et al., 2025) examining long CoTs of these reasoning models have identified a tendency to overthink. Specifically, these models often allocate excessive computational resources—measured in tokens or thinking rounds—to questions that are either extremely simple or for which the answer is already obvious. Our findings align with their observations regarding inefficiency in the inference process.

Setup. As shown in previous experiments, there may not be a significant difference when the operations are just 2 steps apart. For reasoning models, we downsample the operations from 4 to 20, with a 4-step gap between each, using the same 100 samples for each operation.

Recent studies on reasoning models have suggested that providing examples in the prompt can lead to worse performance (OpenAI, 2024b; DeepSeek-AI, 2025), so we directly provide the prompt as described in Section 3. We select two models—QwQ-32B-Preview and DeepSeek-R1—as both make the thinking process visible. For QwQ-32B-Preview, the temperature is set to 0, though temperature setting is not supported for DeepSeek-R1[3].

For QwQ-32B-Preview, significant backtracking and self-validation are frequently observed in nearly every computation. As illustrated in the example below, the model may revisit and revise its steps multiple times when computing one single parameter before arriving at a final, conclusive result. Therefore, we define only the concluding statement of each computation as a single reasoning step. These conclusions often start with phrases such as "so" or "therefore." To detect them systematically, we employ regular expressions.

[2]This also exemplifies redundant reasoning, where two reasoning steps are condensed into a single sentence, with one step being reiterated in the following sentence.

[3]https://api-docs.deepseek.com/guides/reasoning_model

Computing one parameter in long CoTs: Chaoyang's Costco will be 8 + 3 = 1. Is that the answer? Wait, maybe Chaoyang's Costco is affected by Manhattan's Walmart, which is 2 times Chaoyang's Costco. But does this affect the number of Costco in Chaoyang? Not directly, so Chaoyang's Costco will still be 1.

DeepSeek-R1 provides both the reasoning content (i.e., long CoTs) and a summary that adheres to the format requirements, and we evaluate them differently depending on the context. Very rarely (about two problems per operation), the summary will only contain the answer, and we exclude those cases when considering parameter hit and reasoning steps. Since the answers in the summary always match those in the reasoning content (indicating great faithfulness (Lyu et al., 2023)), the accuracy remains the same for both.

For parameter hit, we evaluate the reasoning content and summary separately. Similarly, for reasoning steps, we count them separately—using the same method as QwQ-32B-Preview for the reasoning content and the method outlined in Section 3 for the summary. The token length is the combined total of both the reasoning content and the summary, with the summary typically being less than one-tenth of the length of the reasoning content.

Results. Figure 8 shows that accuracy declines as the number of operations increases. However, we observe that for more complex problems, reasoning models can maintain better performance, with DeepSeek-R1 achieving nearly 60% accuracy for $op = 20$.

Additionally, the parameter hit rate remains high and stable as the problems become more complex. However, we must be cautious, as this may be due to the models simply listing all possible parameters during the long reasoning process. We also observe that in the summary of DeepSeek-R1, some necessary parameters mentioned in the reasoning content are excluded, leading to a slight decrease in the parameter hit rate.

Figure 9 shows that both reasoning steps and token count increase as the number of operations grows. However, unlike previous results, redundant steps continue to increase for QwQ-32B-Preview and the reasoning content of DeepSeek-R1 as the op grows. In contrast, the summary of DeepSeek-R1 introduces little redundancy and even takes fewer steps than necessary for complex problems.

Upon examining the summary, we observe that it often skips reasoning steps that involve implicit dependency, which become more frequent as the number of operations increases. For example, in the correct solution below, the value of *Costco's Product* should be computed as highlighted in red, and since *Costco* only has *Doughnuts*, it equals *Costco's Doughnuts*. However, the summary skips this step.

Correct solution: Define each Walmart 's Banana as A; A = 9. Define each Costco's Doughnuts as V; V = 2 + A = 2 + 9 = 1. Define each Costco 's Product as f; f = V = 1. Define each Walmart 's Pizza as P; P = 9 * f = 9 * 1 = 9.

DeepSeek-R1 summary: Define each Walmart 's Banana as A; A = 9. Define each Costco's Doughnuts as V; V = 2 + A = 2 + 9 = 1. Define each Walmart 's Pizza as P; P = 9 * V = 9 * 1 = 9.

5 Conclusion

This paper reveals a significant decline in model performance as the complexity of math problems increases. We find that more complex CoT prompting is beneficial primarily for simpler problems, offering minimal gains for more complex ones. Reasoning models capable of generating long CoTs can generalize further, but performance still drops as problem complexity increases. These findings highlight the limitations of LLMs in length generalization for math reasoning. We hope this work draws attention to the need for developing models that can learn from shorter examples and apply that knowledge to longer, more complex tasks.

Limitation

In this study, we primarily focus on state-of-the-art open-source models, though the inclusion of closed-source models could serve as a valuable reference. While we aim to provide a comprehensive analysis using various methods and human inspection, it is possible that some aspects worthy of further investigation may have been overlooked.

References

Cem Anil, Yuhuai Wu, Anders Johan Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Samy Jelassi, Stéphane d'Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yuanzhi Li, and François Charton. 2023. Length generalization in arithmetic transformers. *Preprint*, arXiv:2306.15400.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842, Bangkok, Thailand. Association for Computational Linguistics.

Kimi Team. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *Preprint*, arXiv:2501.12599.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. Zebralogic: On the scaling limits of llms for logical reasoning. *Preprint*, arXiv:2502.01100.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *Preprint*, arXiv:2112.00114.

OpenAI. 2024a. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024b. Learning to reason with llms.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown. QWQ-32B-Preview.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness? an analysis of cot in planning. *Preprint*, arXiv:2405.04776.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

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, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2025. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *The Thirteenth International Conference on Learning Representations*.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. 2024. Transformers can achieve length generalization but not robustly. *Preprint*, arXiv:2402.09371.