Length Generalization of Arithmetic Performance

Akash Peddaputha
Tandon School of Engineering
New York University
ap8638@nyu.edu

Ashish Rai
Courant Institute of Mathematical Sciences
New York University
ashish.rai@nyu.edu

Aman Gupta
Courant Institute of Mathematical Sciences
New York University
ag9900@nyu.edu

Abstract

Large language models (LLMs) excel at single-pass NLP tasks like text generation, but struggle with unbounded multistep computations like arithmetic operations with large numbers. This study aims to assess and formulate the length generalization of arithmetic reasoning of LLMs. We observed a significant degradation in model performance when the questions were rephrased with the numerical values scaled in length when tested on the GSM-8K benchmark. We further investigated the scaling behavior on arithmetic tasks and found that state-of-the-art models like GPT-4o-mini and LLaMa-3.1-70B can generate accurate outputs for 4-digit addition and 3-digit multiplication. However, accuracy declines sharply with larger numbers, particularly when the number of unique digits increases. Our results show that while models can generally handle the addition of numbers containing 6 digits and multiplication of 3-digit numbers, they often fail for higher orders. Despite errors on higher-order numbers, we observe a pattern in digit-wise accuracy: the first and the last few digits have higher accuracy than those in the middle, highlighting specific numerical limits in LLM's capabilities for arithmetic tasks. The dataset and the code for reproduction are publicly available.

1 Introduction

Large Language Models (LLMs) have shown remarkable prowess in various domains, such as natural language processing, question answering, and creative tasks [1, 2, 3]. Nevertheless, they still face challenges when it comes to arithmetic operations. Solving math problems necessitates a combination of skills, including text comprehension and arithmetic calculation [4]. Research indicates that the reasoning process employed by LLMs is often based on probabilistic pattern-matching rather than formal logical reasoning [2, 5].

LLMs excel in text generation and summarization, but upon closer examination their limitations in mathematical reasoning capabilities become apparent. In this project, we systematically investigate the arithmetic reasoning of LLMs by examining their generalization performance on elementary arithmetic tasks. We modify numerical values in the GSM-8K benchmark [6] to assess the sensitivity of models like GPT-4o-mini and LLaMa-3.1-70b-Instruct to increasing numerical complexity. Additionally, we explore the scaling behavior of LLMs on these tasks by gradually increasing the complexity of numerical values, testing their accuracy on operations from 1-digit to 7-digit numbers.

Arithmetic Length Generalization Dataset (ALGD, 2024)
https://github.com/raishish/arithmetic-interp
DS-GA 1011 Natural Language Processing

Our experiments show that LLMs can handle simpler arithmetic tasks but their accuracy deteriorates with increasing complexity, especially for smaller numbers with higher unique digit counts.

Through this study, we aim to understand the impact of numerical complexity and arithmetic operations on the scaling behavior of LLMs, rather than improving arithmetic operations.

2 Related Work

Recent advances in LLMs have spurred interest in improving their mathematical reasoning capabilities. [2] introduces GSM-Symbolic, an improved benchmark to evaluate LLMs on math reasoning tasks. Their approach reveals the fragility of LLM reasoning, with model performance sensitive to slight numerical changes. [7] further explores LLM reasoning mechanisms by identifying critical attention heads and MLPs essential for arithmetic calculations.

[8] addresses scaling behavior of LLMs across numeral systems, comparing base-10 and base-100 tokenization schemes. [9] provides a survey on LLMs in mathematical reasoning, categorizing problem types and identifying relevant datasets for arithmetic, geometry, and theorem proving. Finally, the work by [10] investigates transformers' length generalization capabilities. It focuses on transformer architectural aspects that affect their generalization on length-variable tasks.

3 Approach

We evaluated the model's performance on the GSM-8K dataset by multiplying numerical values by multiples of 10, which did not increase the complexity much since the number of unique digits remains constant. The model consistently produced correct answers for such questions. To further increase complexity, we introduced random numbers with more unique digits and tested the model's handling of these variations. We also used one-shot and few-shot and Chain of Thought (CoT) prompting to improve multi-step reasoning. The models were prompted to not use or generate code to generate responses, thus avoiding the impact of code interpreting techniques. We experimented with different sampling settings and conducted repeated tests under similar conditions to ensure consistency.

To mechanistically understand how responses are generated for arithmetic operations, we use transformer interpretability techniques to look for attribution of intermediate layers of the network to the final response.

4 Experiments

4.1 Data

In this work, we evaluate our models using two datasets: the widely-used GSM-8K test dataset and a custom dataset we created for evaluating length generalization of arithmetic performance.

GSM-8K Dataset: For evaluating our models' mathematical reasoning capabilities, we use the GSM-8K dataset [6]. GSM-8K contains grade-school-level math problems designed to assess logical reasoning and arithmetic capabilities in large language models (LLMs).

Arithmetic Length Generalization Dataset (ALGD): To evaluate our model's performance on elementary arithmetic operations, we designed a custom dataset with 3,500 examples for addition, multiplication, subtraction, division, and modulus. Each operation includes 100 examples with 1-digit to 7-digit numbers (around 700 per operation), allowing us to assess the model's capacity to handle progressively more complex scenarios.

For both datasets, the task involves generating an accurate numerical output in response to the arithmetic input or question.

4.2 Evaluation method

We assess the model's arithmetic reasoning using accuracy metric on the GSM-8K and ALGD datasets. For GSM-8K, accuracy measures the percentage of correctly answered questions, benchmarking general mathematical reasoning and comparing with prior work. On ALGD, we measure digit-wise accuracy separately across digit complexities for all operations, evaluating the model's scalability and robustness with numerical complexity along with overall accuracy. This comprehensive approach highlights the model's strengths and limitations in diverse mathematical tasks.

4.3 Experimental details

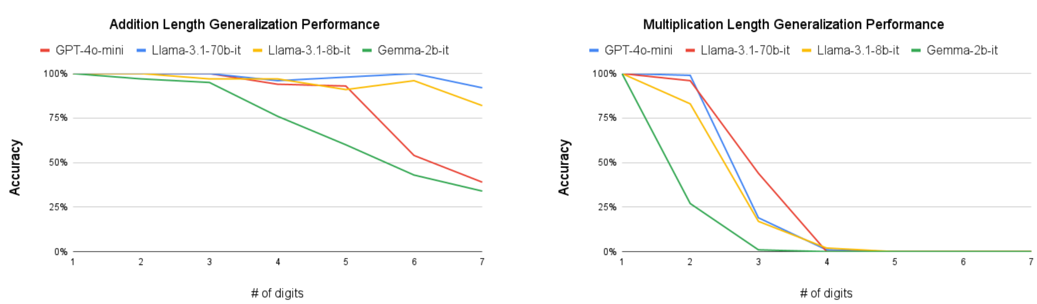
We use the configurations defined in Table 1 to evaluate GPT-4o-mini, Llama-3.1-70b-Instruct, Llama-3.1-8b-Instruct, and gemma-2b-it on both GSM-8K and ALGD datasets. We explicitly specify in the system prompt to not use any code to avoid GPT-4o-mini from using its Code Interpreter feature. The exact prompt for different models is given in the Appendix section 8.

We first compare the performance of both models on GSM-8K to establish a baseline. Inspired by GSM-Symbolic [2], we then increased the number of digits to increase the complexity of the questions and to move away from the training distribution. Finally, we evaluate the models on our generated ALGD dataset.

We use Tuned Lens [11], built upon the Logit Lens [12] approach, to project the activations from intermediate layers of Llama-3.1-8B-Instruct into the model's vocabulary space. Tuned Lens helps avoid misalignment observed in Logit Lens due to evolution of representations by learning layer-specific probes to minimize the distribution shift.

4.4 Results

It can be seen from Figure 4 that frontier models perform with $\sim$94% accuracy on the GSM-8K dataset. However, when the number of digits in the numerical values are increased, our observations reveal that models perform well with numbers containing fewer unique digits, such as 480000, but their accuracy declines with numbers that have more unique digits, even on a small scale (e.g., 48346) 6.2. Additionally, the models often misrepresent high-magnitude numbers, sometimes interpreting quadrillions as billions or using inconsistent scientific notation. As numeric complexity increases, models occasionally "hallucinate" numbers, particularly in responses to simple arithmetic prompts involving large sums. When using Few-Shot prompting, models show an over-fitting tendency to the set number of reasoning steps, producing accurate intermediate calculations but failing to combine these results correctly in the final answer. With Chain-of-Thought prompting, while models can decompose the operation between large numbers into those with smaller numbers and generate accurate results, they struggle with summing these intermediate results accurately.



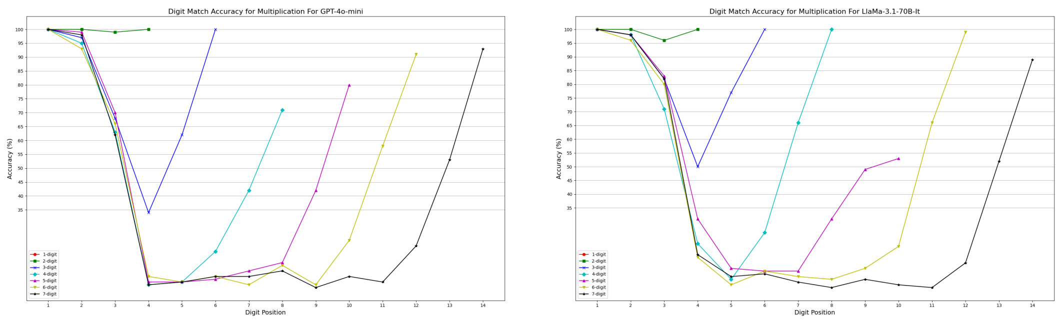
(a) (b)

Figure 1: **Performance deteriorates with increase in number of digits**. **a.** Llama-3.1-70b-Instruct performs better than GPT-4o-mini whose performance drops after 5-digit addition (b) Performane of both Llama-3.1-70b-Instruct and GPT-4o-mini reaches 0% after 3-digit multiplication.

We observe performance of addition 1a and multiplication 1b limiting to 6-digit and 3-digit numbers, respectively, but exhibit significant errors beyond these limits. Furthermore, our results show that while models can generally handle computations involving numbers up to 6 digits, they often fail for higher orders, with larger unique digits in smaller numbers exacerbating error rates. One interesting observation is that, the models often generate the first and last few digits accurately, even as overall accuracy declines 2a 2b. This could be related to frequencies of combination of specific digits in the training corpus or specific tokenization methods. More graphs can be seen in Appendix sections

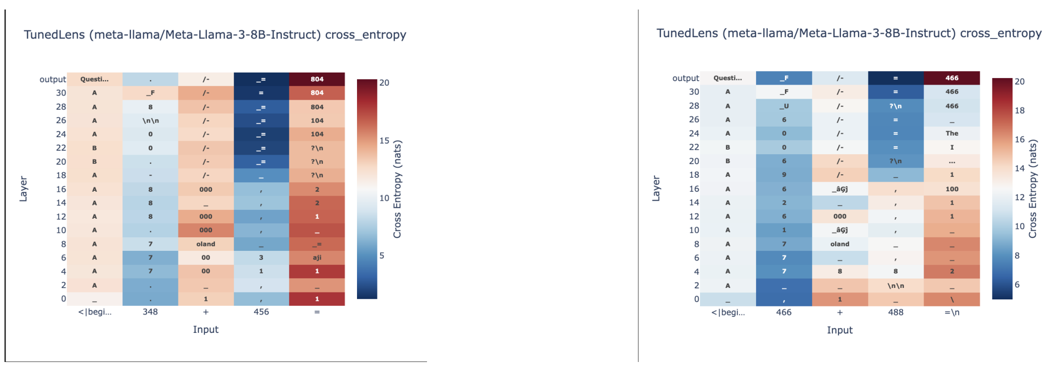
5 and 6. We plan to explore further in this direction. The trend remains the same irrespective of different training recipes and tokenization strategies across the two models.

With tuned lens, it can be seen that the final numerical answer is generated in the later layers of Llama-3.1-8B 3a. In case of multi-token answers, even after providing the first token in the prompt, the model fails to generate the next token 3b.



(a) (b)

Figure 2: **Digit accuracy in prediction of multiplication of two 7-digit numbers.** The trend remains the same irrespective of different training recipes and tokenization strategies across the two models.



(a) (b)

Figure 3: **Tuned Lens Plots for Cross Entropy Loss**. (a) Llama-3.1-8b-Instruct model output at every layer for addition (b) Llama-3.1-8b-Instruct model output at every layer for multiplication

5 Future Work

Following up on this work, We aim to explore the underlying reasons behind the observed limitations in arithmetic reasoning by conducting white-box experiments. This will involve intermediate decoding to analyze model outputs at different stages, as well as applying mechanistic interpretability methods to identify specific components or patterns influencing performance. These insights will help us understand the model's internal processing of arithmetic tasks and inform potential improvements.

References

[1] Apple Intelligence Team. Apple intelligence foundation language models, 2024.

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

[3] OpenAI. Gpt-4 technical report, 2024.

[4] Andreas Opedal, Alessandro Stolfo, Haruki Shirakami, Ying Jiao, Ryan Cotterell, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. Do language models exhibit the same cognitive biases in problem solving as human learners?, 2024.

[5] Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J. Su, Camillo J. Taylor, and Dan Roth. A peek into token bias: Large language models are not yet genuine reasoners, 2024.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. 2021.

[7] Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation. *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[8] Zhejian Zhou, Jiayu Wang, Dahua Lin, and Kai Chen. Scaling behavior for large language models regarding numeral systems: An example using pythia. *arXiv preprint arXiv:2409.17391*, 2024.

[9] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

[10] Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *Proceedings of ICLR 2024*, 2024.

[11] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

[12] nostalgebraist. Logit lens. LessWrong, 2022. Blog post.

6 Appendix

6.1 Performance of GPT-4o-Mini and Llama-3.1-70B-instruct Models on GSM8K dataset

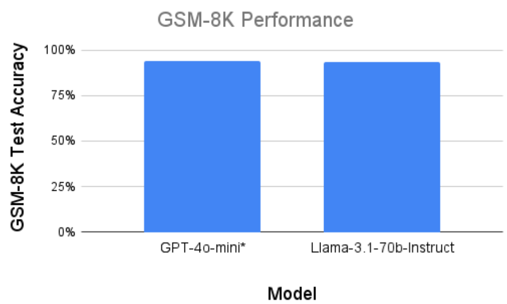


Figure 4: **Frontier models like GPT-4o-mini and Llama-3.1-70b-Instruct score >90% on mathematical reasoning datasets like GSM-8K.** *Due to the constraint max_completion_tokens set to 512, 6 out of 1319 responses by GPT-4o-mini were truncated. Accuracy is calculated after removing these question-answer pairs.

6.2 GSM-8K Scaling Experiments with Llama-3.1-70B-Instruct

6.2.1 No Scaling

Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
Correct Answer: 72

Model Response: In April, Natalia sold 48 clips. In May, she sold half of those, so she sold 48 / 2 = 24 clips. To find the total number of clips sold in both months, we need to add the number of clips sold in each month.

Total clips sold in April and May = Clips sold in April + Clips sold in May
= 48 + 24
= **72**

6.2.2 Scaled Variation 1 - multiply with 1000

Question: Natalia sold clips to 480000 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
Correct Answer: 720000

Model Response: In April, Natalia sold 480000 clips. In May, she sold half of those, so she sold 480000 / 2 = 240000 clips.

To find the total number of clips sold in both months, we need to add the number of clips sold in each month.

Total clips sold in April and May = Clips sold in April + Clips sold in May
= 480000 + 240000
= 720000

6.2.3 Scaled Variation 2 - Increase unique digits

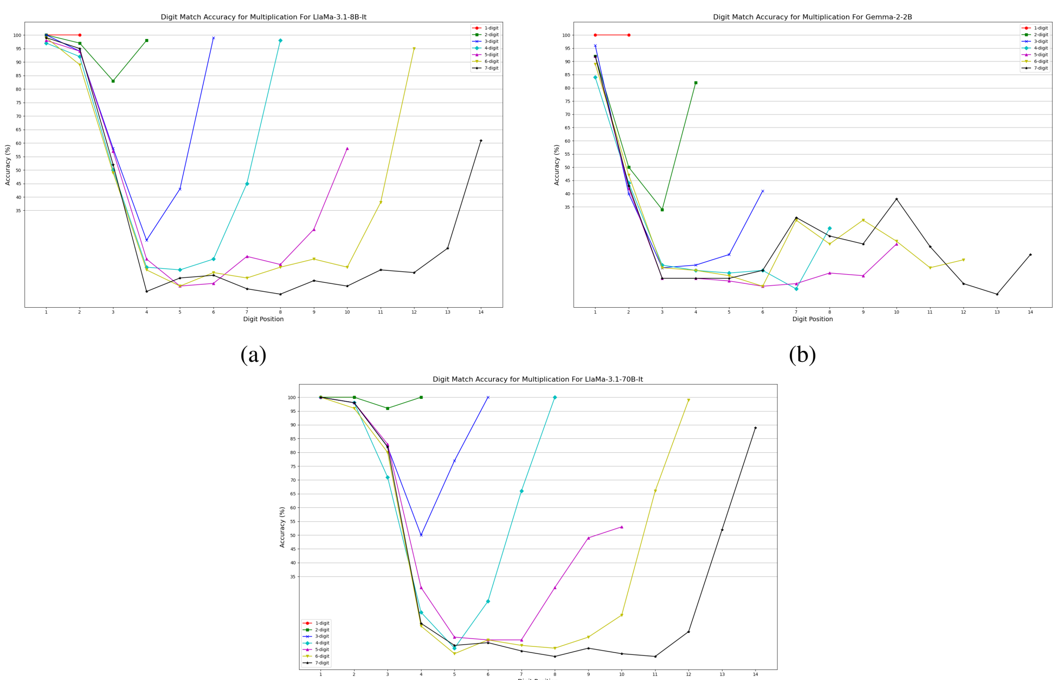
Question: Natalia sold clips to 48346 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
Correct Answer: 72519

Model Response: In April, Natalia sold 48346 clips.

If she sold half as many clips in May, that would be 48346 / 2 = **24183** (**24173**).

So, the total number of clips Natalia sold is 48346 + 24183 = **72529**.

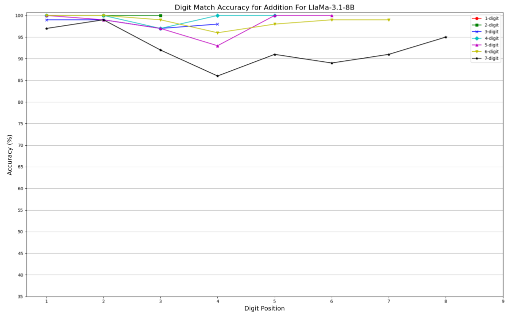
6.3 Digit Matching results for multiplication



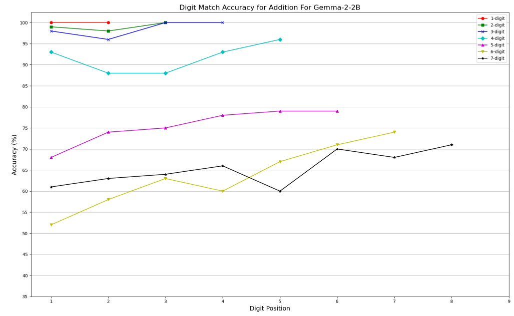
(a) (b) (c)

Figure 5: **Performance deteriorates with increase in number of digits**. **a.** Performance of Llama-3.1-8b-Instruct **b.** Performance of Gemma-2-2B **c.** Performance of Llama-3.1-70B-Instruct

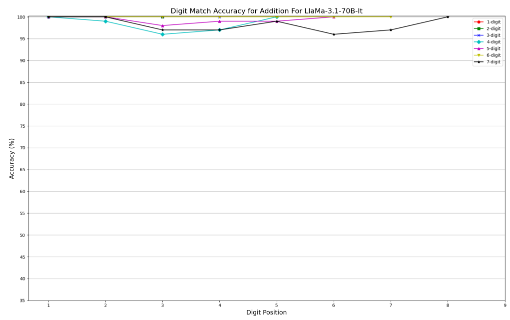
6.4 Digit Matching results for addition



(a)



(b)



(c)

Figure 6: **Performance deteriorates with increase in number of digits**. **a.** Performance of Llama-3.1-8b-Instruct **b.** Performance of Gemma-2-2B **c.** Performance of Llama-3.1-70B-Instruct

7 Model Inference Configurations

Table 1: **Model parameters used for generating responses**

Parameter	**Description**	**Value**
max_completion_tokens	Max # of completion tks including output and reasoning tks	512
temperature	Sampling temperature	0.7
top_p	Probability mass to sample tokens from	0.9
n	# of sequences generated	1
seed	Seed for random number generator	0
system_prompt		You are a helpful assistant. Do not use or generate code in your responses.

8 Prompts used for generating responses

GPT-4o-mini Prompt

```
messages=[
    {   "role": "system",
        "content": "You are a helpful assistant. Do not use or
           generate code in your responses. Please respond with only
           the final answer, formatted as a single number without
           additional explanation or context."},
    {   "role": "user",
        "content": f"{query}\n\nPlease provide only the final answer
           as a single number."}
    ]
```

For Gemma and Llama Models:

```
messages=[
    {   "role": "system",
        "content": "You are a helpful assistant. Do not use or
           generate code in your responses. Also append the final
           numerical answer on a new line append with '#### '"},
    {   "role": "user",
        "content": query}
    ]
```