AI-RITHMETIC*

Alex Bie, Travis Dick, Alex Kulesza, Prabhakar Raghavan, Vinod Raman, Sergei Vassilvitskii
Google

ABSTRACT

Modern AI systems have been successfully deployed to win medals at international math competitions, assist with research workflows, and prove novel technical lemmas. However, despite their progress at advanced levels of mathematics, they remain stubbornly bad at basic arithmetic, consistently failing on the simple task of adding two numbers. We systematically investigate this phenomenon. We demonstrate empirically that all frontier models suffer significantly degraded accuracy for integer addition as the number of digits increases. Furthermore, we show that most errors made by these models are highly interpretable and can be attributed to either operand misalignment or a failure to carry correctly. These two error classes explain 87.9%, 62.9%, and 92.4% of Claude Opus 4.1, GPT-5, and Gemini 2.5 Pro errors, respectively. We show that misalignment errors can be related to tokenization, and that carrying errors appear largely as independent random failures.

1 INTRODUCTION

Large language models have won medals in international mathematics competitions (Luong and Lockhart, 2025; Wei, 2025), have been used to find new lemmas and improve existing proofs in research mathematics (Gans, 2025; Novikov et al., 2025; Nagda et al., 2025; Georgiev et al., 2025), and have been adopted to solve open problems (Fountoulakis, 2025; Sellke and Yin, 2025). At the same time, it is widely known (and we confirm here) that these systems are remarkably unreliable at basic arithmetic. This is not inherently contradictory: arithmetic is different from advanced mathematics in many ways. But for humans, these skills are both clearly related and strongly ordered, arithmetic being a simple but critical part of the foundation on which deeper mathematics is built. Therefore, it may seem surprising that AI systems can outperform experts at more "advanced" tasks but fail to keep pace with kindergartners in the "simple" ones.

Accepting that AI systems do not exhibit human-like patterns of performance, we might instead imagine them as algorithmic machines performing complex behaviors by composing simple ones (as in, for example, long addition). Indeed, this is the idea behind the current wave of *reasoning models* trained to perform a series of intermediate operations to reach a final answer. But this compositional view also fails to describe arithmetic performance, which we find is not even monotonic: AI systems are sometimes better at adding longer numbers than shorter ones. Moreover, while bigger models are generally better, all current models make significant numbers of mistakes as the problem size grows. This suggests that scaling alone is not likely to solve the problem, despite fact that the model only needs to learn and orchestrate handful of simple behaviors to solve the problem for all lengths.

Arithmetic, then, is an interesting case study in how modern AI systems can confound intuitions. Here, we systematically investigate model behavior on simple integer addition problems and characterize the observed mistakes. We find that errors are generally explainable in intuitive terms, offering a view of AI systems that is neither strictly human-like nor purely compositional, but idiosyncratic and dependent on design choices such as tokenization and auto-regression. While tool use or task-focused training may be used to address the specific arithmetic difficulties we discuss here, the larger point is that today's models still have considerable weaknesses that we cannot fully anticipate. Manually correcting individual deficiencies as they are discovered is not a viable path toward reliable general-purpose systems. We use arithmetic as a convenient probe to understand today's AI failures and illuminate larger problems that have yet to be solved.

*Authors are listed in alphabetical order. Correspondence to {alexbie, tdick, kulesza, pragh, vinodraman, sergeiv}@google.com.

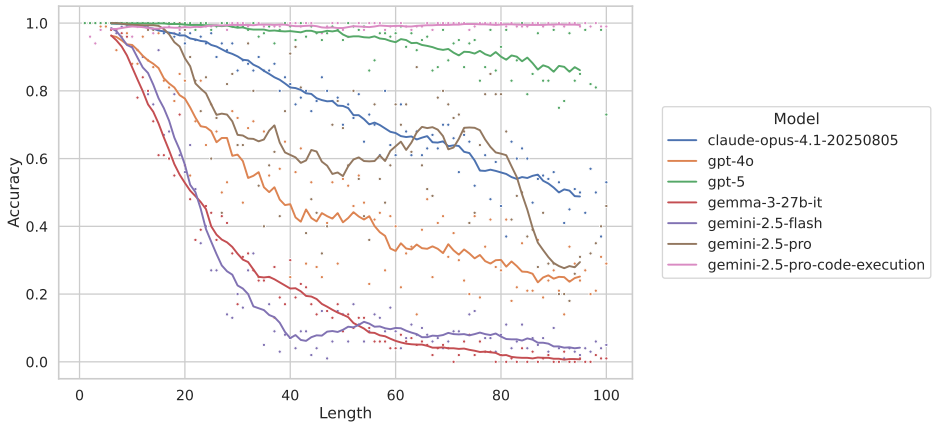

Figure 1: **Frontier models fail at adding long numbers.** The plotted points show the accuracy of the tested models on addition problems at each length. Since model performance fluctuates significantly, we also include a centered 10-point moving average curve.

1.1 Summary of Findings

- When adding two numbers, *all* frontier models perform poorly as the length of the numbers increases (Figure 1). However, performance is not necessarily monotone in length.
- Most observed mistakes are due either to *misalignment* or *close carry* errors (Figure 2). *Misalignment* errors occur when one of the input operands is incorrectly shifted. *Close carry* errors occur when the model incorrectly carries or fails to carry in borderline cases (that is, where the sum of digits is 9 but the model still carries a 1, or the sum is 10 and the model fails to carry). Misalignment and close carry errors comprise 87.9%, 62.9%, and 92.4% of Claude Opus 4.1, GPT-5, and Gemini 2.5 Pro errors, respectively.
- Further investigation into error classes reveals that:
  - Misalignment errors are often periodic with respect to argument length, a fact that may be explained by tokenization, especially for models that use multi-digit tokens.
  - The occurrence of close carry errors is largely consistent with an *independent error model* where models err on each close carry independently with probability p. If an addition problem contains n close carries, this predicts a $(1-p)^n$ chance of getting the problem correct (i.e., the LeCun (2023) model), and a geometrically distributed first error position.

1.2 Related Work

Although language models can perform arithmetic tasks with non-trivial accuracy in certain settings (Wei et al., 2022a; Yang et al., 2023; Maltoni and Ferrara, 2024), their failure to generalize and overall poor performance have been widely observed (Saxton et al., 2019; Nogueira et al., 2021; Dziri et al., 2023; Qian et al., 2023; Gambardella et al., 2024; Testolin, 2024; Yan et al., 2025; Loeber, 2024). Recent work has focused on understanding how AI systems represent numbers or implement arithmetic operations (Stolfo et al., 2023; Dziri et al., 2023; Nikankin et al., 2025; Deng et al., 2024; Zhang et al., 2024; Baeumel et al., 2025b; Levy and Geva, 2025; Kantamneni and Tegmark, 2025), as well as how they can be improved. Techniques include prompting or training with chain-of-thought (Wei et al., 2022b; Lee et al., 2023), augmenting the model's abilities with symbolic systems (Yang et al., 2024; Dugan et al., 2024), adding hints or restructuring the problem (Nogueira et al., 2021; Zhou et al., 2023), increasing numerical precision (Feng et al., 2024), and utilizing improved positional encodings (Shen et al., 2023; McLeish et al., 2024; Zhou et al., 2024).

Singh and Strouse (2024) find arithmetic errors to be highly *position-dependent* rather than problem difficulty-dependent. Baeumel et al. (2025a) analyze carrying errors, attributing them to the inability of models to anticipate if carries will cascade left. They study multi-operand, 3-digit addition on smaller models, while we focus on two-operand, n-digit addition in frontier reasoning models, leading us to uncover the "local" nature of close carry errors, where they can be modeled as independent errors occurring with probability p.

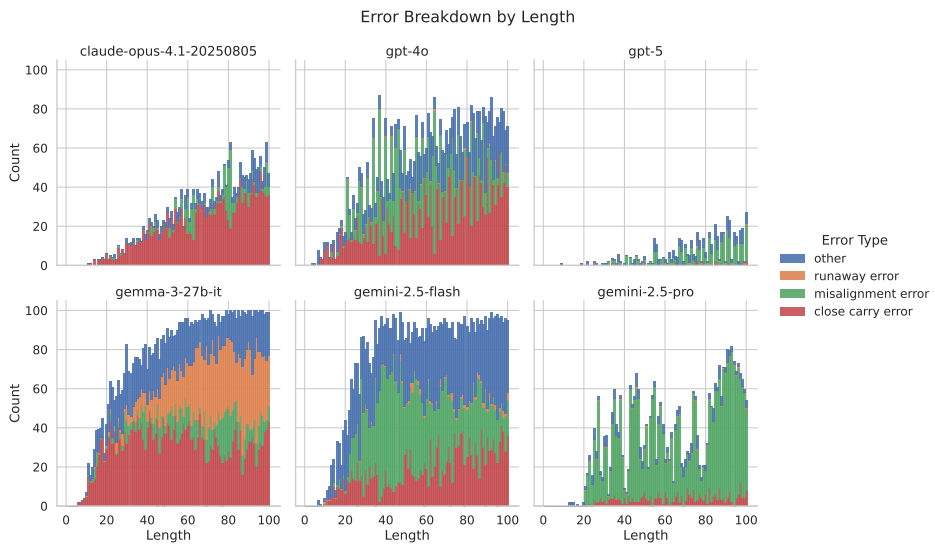

Figure 2: **Most errors can be explained**. We plot the distribution of error types at each length. For a description of error types, see Section 3. We find that close carry and misalignment errors cover 87.9% of Claude Opus 4.1's mistakes, 62.9% of GPT-5's mistakes, 92.4% of Gemini 2.5 Pro's mistakes, and at least 55.6% of every model's mistakes.

2 Model Performance

In this section, we investigate the performance of frontier LLMs when adding two numbers using prompts of the form:

`What is` A `+` B`? Write just the answer.`[1]

Here A and B are uniformly generated d-digit integers. For more details of the experimental setup see Appendix A.

Results. Figure 1 shows the fraction of correct responses a function of the length d. With the exception of Gemini 2.5 Pro Code Execution, the accuracy of all models drops significantly below 100% as the number of digits increases. Even at a moderate length of 20, most models exhibit a nontrivial error rate. This suggests that the varieties of architectures, data mixtures, and training practices used by current models are not sufficient for learning to accurately perform basic addition. Gemini 2.5 Pro Code Execution's stronger performance can be attributed to the fact that it usually calculates the sum using short python scripts. Nonetheless, we still observe errors when the model fails to use the tool or fails to correctly copy the answer.

3 Exploration of Mistakes

We explore the types of mistakes evident in Figure 1. We find that a significant fraction of mistakes can be explained intuitively. Recall that when adding a pair of numbers, the canonical long addition algorithm aligns the operands vertically and then computes a sequence of digit sums, one for each column, proceeding from least to most significant position. When a column sum exceeds 9, a one is "carried" to the next column.

Most model mistakes are consistent with one of two natural failure modes during the execution of this algorithm: (1) misalignment, where the digits of one operand are shifted by a few positions in either direction, and (2) close carry failure, where a carry is performed in error when the column sum is 9, or not performed when the column sum is 10. These two types cover 92.4% of Gemini 2.5 Pro's mistakes, 87.9% of Claude Opus 4.1's mistakes, and at least 55.6% of every model's mistakes.

[1] We also tested several other prompts, but did not find any that elicited significantly better accuracy. We argue that an intelligent system should not depend meaningfully on the form of the prompt as long as the intent is clear.

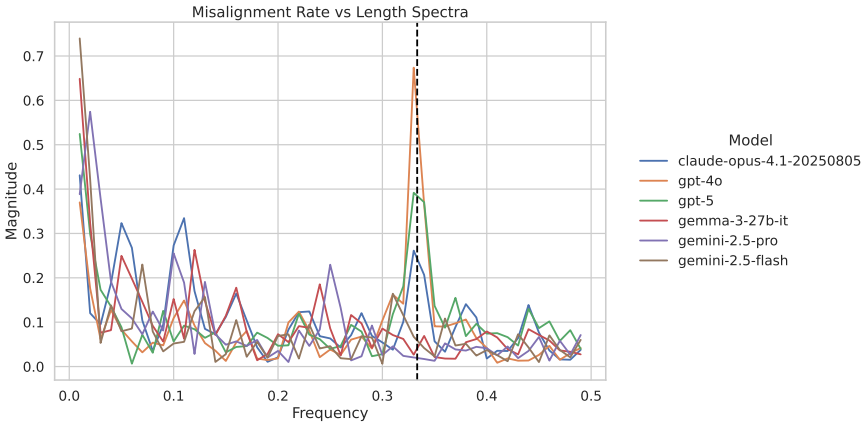

Figure 3: Spectra of the misalignment rate vs. length for each model. Each spectrum is normalized to have ℓ_2 norm. Models employing 3-digit tokenization (Claude Opus 4.1, GPT-5) exhibit a noticeable spike at $1/3$.

Misalignment errors. We say that a model's extracted answer for the sum of A and B contains a *misalignment error* if at least the first 6 digits of the extracted answer match the sum when adding A and B with the arguments offset by up to 10 digits in either direction. We limit the test to six digits because models tend to compound their errors as they go, making long misaligned matches unlikely. We require at least six digits to avoid false detections. For details and examples see Appendix B.

Close carry errors. We define close carry errors as those where: (a) the k-th position digit is too large by 1 and the column to the right has a sum of 9; or (b) the k-th position digit is too small by 1 and the column to the right has a correct sum of 10. We give a more precise definition in Appendix B.

Runaway errors. In addition to misalignment and close carry errors, we observe that some of the models frequently output answers that are much longer than the correct answer (sometimes thousands of digits long). We say that an extracted answer is a *runaway* error if its length is at least 50% longer than the true answer.

Results. Figure 2 shows how the mistakes made by each model break down into aforementioned mistake types. Since the error types we consider are not mutually exclusive, we classify a model's response according to the first mistake type it matches in the following order: runaway, misalignment, close carry. Any mistakes that do not match the conditions for these three error types are classified as "other". For Claude Opus 4.1, Gemini 2.5 Pro, GPT-4o, and GPT-5, the majority of their mistakes match at least one of the three mistake types. Most of the mistakes made by Claude Opus 4.1 are close carrying errors, while most of the mistakes made by Gemini 2.5 Pro and GPT-5 are misalignment errors.

4 Explanation of Mistakes

Tokenization and misalignment. In Figure 3 we plot the spectrum of the misalignment error rate as a function of length. Pronounced spikes at $1/3$ are observed for models employing 3-digit tokenization (Claude 4.1 Opus, GPT-4o, GPT-5). Meanwhile, we see no spikes at $1/3$ for the other models, in which we find strong evidence that they tokenize digits individually.[2] For more details, please see Appendix D.1.

These results suggest that misalignment errors may be exacerbated when operands are encoded with tokens containing variable numbers of digits. However, we also note that models using single-digit tokenizers tended to exhibit more misalignment errors overall, so it may be that longer tokens actually help to *reduce* misalignment errors, and this effect is muted when mixed-length tokens are used.

[2]Gemma 3's tokenizer tokenizes digits individually (Google, 2025). For evidence that Gemini 2.5 tokenizes digits individually: Gemma Team (2025) reports using the same tokenizer as Gemini 2; and the `count_tokens` API for Gemini 2.5 Pro and Flash reports an increase of d tokens when passed a d-digit number.

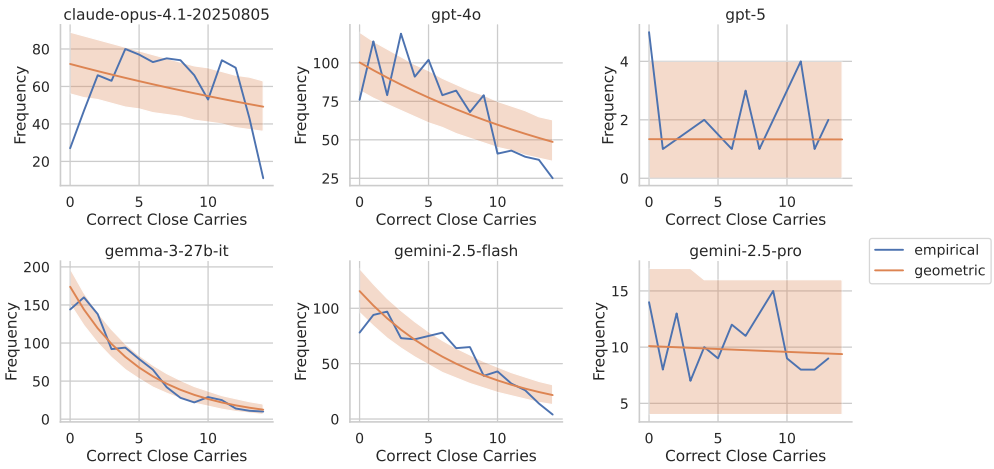

Figure 4: The number of correct close carries observed prior to the first close carry error. We compare to a geometric model that assumes the LLM fails on each close carry independently with probability p; this predicts exactly i correct carries with probability $p(1-p)^i$. The shaded region shows the 95% confidence interval for geometric model.

Independence and close carry errors. Close carry errors are "local" – they produce small overall edit distance between the model output and the correct answer (Figure 5). We propose a simple one-parameter stochastic model of close carry errors, and show that it often matches the observed patterns of mistakes. Assume a given addition problem involves n close carries. We imagine that the model proceeds from left to right, producing the correct output digit at each position that does not involve a close carry. Upon reaching a close carry position, the model flips a coin and makes a close carry error with probability p. If it does not make an error, it continues, and after reaching the next close carry position, flips another independent coin with the same bias p. The model will ultimately produce a correct answer with probability $(1-p)^n$. For more details on the stochastic model, see Section D.2.

We assess whether this stochastic process is representative: we target $n = 15$ close carries and then select only addition problems with exactly n close carries. For each model, we set p so that the predicted error matches the observed error rate on these problems, including only problems for which the model either made a close carry error or was correct. We then examine, for each selected problem, the number of *successful* close carries before the first carry error, comparing the distribution observed empirically with the one predicted by the stochastic model. As shown in Figure 4, the two distributions are often closely (though not perfectly) aligned. These results are consistent with the idea that some models make close carry errors in a simple, independent way, and do not correlate errors across positions.

5 Discussion

Recent work has studied the representational power of Transformers (Weiss et al., 2021; Schnabel et al., 2025; Sanford et al., 2024). While the specific modeling choices vary, arithmetic generally falls in the class of tasks that can be solved exactly. In fact, addition is one of the examples used to explain how to "think like a transformer" (Weiss et al., 2021).

However, our evidence suggests that modern LLMs are not finding reliable implementations for addition, instead resorting to error-prone strategies that give poor results and exhibit various idiosyncrasies. Bigger models tend to perform better, but are still easily dominated by a simple calculator, especially as the number of digits grows beyond 50. If we expect intelligent systems to perform basic arithmetic without falling back on calculator tools, then we will need new model designs, data sources, and/or training strategies.

Of course, arithmetic is itself just an example. Success at arithmetic is easy to measure, and we have ready expectations about how an intelligent system ought to perform. But today's AI systems fail in many unique ways, some of which may be similarly easy to characterize and understand, and others which may be hard to delineate or even see clearly at all. Perhaps LLMs are simply not the right tool for tasks where one must identify a single correct answer. Or perhaps even "soft" probabilistic reasoning is too much to expect in complex settings. But we will need a clearer understanding of these boundaries and how to move beyond them if we want robust AI deployments that meet our expectations.

REFERENCES

Anthropic. Token counting. https://platform.claude.com/docs/en/build-with-claude/token-counting, 2025. Accessed on Jan. 28, 2026.

Tanja Baeumel, Josef Van Genabith, and Simon Ostermann. The lookahead limitation: Why multi-operand addition is hard for LLMs. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 250–262, Suzhou, China, November 2025a. Association for Computational Linguistics. doi: 10.18653/v1/2025.blackboxnlp-1.15. URL https://aclanthology.org/2025.blackboxnlp-1.15/.

Tanja Baeumel, Daniil Gurgurov, Yusser al Ghussin, Josef van Genabith, and Simon Ostermann. Modular arithmetic: Language models solve math digit by digit. *arXiv preprint arXiv:2508.02513*, 2025b.

Chunyuan Deng, Zhiqi Li, Roy Xie, Ruidi Chang, and Hanjie Chen. Language models are symbolic learners in arithmetic. *arXiv preprint arXiv:2410.15580*, 2024.

Owen Dugan, Donato Jiménez-Benetó, Charlotte Loh, Zhuo Chen, Rumen Dangovski, and Marin Soljacic. Occamllm: Fast and exact language model arithmetic in a single step. *Advances in Neural Information Processing Systems*, 37: 35665–35699, 2024.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36:70293–70332, 2023.

Guhao Feng, Kai Yang, Yuntian Gu, Xinyue Ai, Shengjie Luo, Jiacheng Sun, Di He, Zhenguo Li, and Liwei Wang. How numerical precision affects mathematical reasoning capabilities of llms. *arXiv preprint arXiv:2410.13857*, 2024.

Kimon Fountoulakis. Gpt-5.2 solves our colt 2022 open problem: "running time complexity of accelerated l1-regularized pagerank" using a standard accelerated gradient algorithm and a complementarity margin assumption. https://x.com/kfountou/status/2000957773584974298, 2025.

Andrew Gambardella, Yusuke Iwasawa, and Yutaka Matsuo. Language models do hard arithmetic tasks easily and hardly do easy arithmetic tasks. *arXiv preprint arXiv:2406.02356*, 2024.

Joshua S. Gans. The efficient market hypothesis when time travel is possible. *Economics Letters*, 248:112209, 2025. ISSN 0165-1765. doi: https://doi.org/10.1016/j.econlet.2025.112209. URL https://www.sciencedirect.com/science/article/pii/S0165176525000461.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale, 2025. URL https://arxiv.org/abs/2511.02864.

Google. Tokenizer. https://gemma-llm.readthedocs.io/en/latest/colab_tokenizer.html, 2025. Accessed on Jan. 28, 2026.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*, 2025.

Yann LeCun. Unpopular opinion about ar-llms. https://x.com/ylecun/status/1640122342570336267, 2023.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

Amit Arnold Levy and Mor Geva. Language models encode numbers using digit representations in base 10. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 385–395, 2025.

John Loeber, 2024. URL https://loeber.substack.com/p/16-notes-on-arithmetic-in-gpt-4.

Thang Luong and Edward Lockhart, 2025. URL https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/.

Davide Maltoni and Matteo Ferrara. Arithmetic with language models: From memorization to computation. *Neural Networks*, 179:106550, 2024.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, et al. Transformers can do arithmetic with the right embeddings. *Advances in Neural Information Processing Systems*, 37:108012–108041, 2024.

Ansh Nagda, Prabhakar Raghavan, and Abhradeep Thakurta. Reinforced generation of combinatorial structures: Applications to complexity theory, 2025. URL https://arxiv.org/abs/2509.18057.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=O9YTt26r2P.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*, 2021.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

OpenAI. Tokenizer. https://platform.openai.com/tokenizer, 2025a. Accessed on Jan. 28, 2026.

OpenAI. Tiktoken. https://github.com/openai/tiktoken, 2025b. Release 0.12.0, Accessed on Nov. 3, 2025.

Jing Qian, Hong Wang, Zekun Li, Shiyang Li, and Xifeng Yan. Limitations of language models in arithmetic and symbolic induction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9285–9298, 2023.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth, 2024. URL https://arxiv.org/abs/2402.09268.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Tobias Schnabel, Kiran Tomlinson, Adith Swaminathan, and Jennifer Neville. Lost in transmission: When and why llms fail to reason globally, 2025. URL https://arxiv.org/abs/2505.08140.

Mark Sellke and Steven Yin. On learning-curve monotonicity for maximum likelihood estimators, 2025. URL https://arxiv.org/abs/2512.10220.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*, 2023.

Aaditya K. Singh and DJ Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier llms, 2024. URL https://arxiv.org/abs/2402.14903.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. *arXiv preprint arXiv:2305.15054*, 2023.

Alberto Testolin. Can neural networks do arithmetic? a survey on the elementary numerical skills of state-of-the-art deep learning models. *Applied Sciences*, 14(2):744, 2024.

Alexander Wei. "1/N I'm excited to share that our latest @OpenAI experimental reasoning LLM has achieved a longstanding grand challenge in AI: gold medal-level performance on the world's most prestigious math competition—the International Math Olympiad (IMO).". https://x.com/alexwei_/status/1946477742855532918, 2025.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022a. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

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

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers, 2021. URL https://arxiv.org/abs/2106.06981.

Yang Yan, Yu Lu, Renjun Xu, and Zhenzhong Lan. Do large language models truly grasp addition? a rule-focused diagnostic using two-integer arithmetic. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13478–13494, 2025.

Xiaocheng Yang, Bingsen Chen, and Yik-Cheung Tam. Arithmetic reasoning with llm: Prolog generation & permutation. *arXiv preprint arXiv:2405.17893*, 2024.

Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. Gpt can solve mathematical problems without a calculator. *arXiv preprint arXiv:2309.03241*, 2023.

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. *arXiv preprint arXiv:2409.01659*, 2024.

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. *arXiv preprint arXiv:2310.16028*, 2023.

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

Response	Correct / Incorrect
`1234`	`correct`
`1,234`	`correct`
`The answer is 1234.`	`correct`
`Maybe the answer is 1234 but really it is 9999.`	`correct`
`9991234000`	`incorrect`
`1 234`	`incorrect`
`5678`	`incorrect`

Table 1: Examples of correct and incorrect model responses when the sum is `1234`

offset s	$A \cdot 10^{\max(0,s)}$	$B \cdot 10^{\max(0,-s)}$	$\text{offsetsum}(A, B, s)$	**Required Prefix**
-2	555555	12345600	12901155	`129011`
-1	555555	1243560	1790115	`179011`
1	5555550	123456	5679006	`567900`
2	55555500	123456	55678956	`556789`

Table 2: This table shows the collection of 6-digit prefixes that would count as a misalignment error for the sum of $A = 555555$ and $B = 123456$. Each row corresponds to one offset $s \in \{-2, -1, 1, 2\}$ and includes the offset arguments, the offset sum, and the prefix a model answer must have to be considered a misalignment mistake. This table shows the example for offsets up to 2 digits in either direction, but our implemented test searches for offsets of up to 10 digits.

A Experimental Setup

For each length $d \in \{1, \ldots, 100\}$, we generate 100 prompts by setting A and B to be d-digit numbers chosen independently and uniformly at random from $\{10^{d-1}, \ldots, 10^d - 1\}$. We send each prompt to each model and collect their responses.

To extract an answer from the model's response, we remove all commas (to account for different numerical formatting conventions) and then extract the span of digits with the lowest edit distance to the correct sum $A + B$. We say that the model is correct if the match is exact, and incorrect otherwise. This test is lenient: if the model response includes multiple spans of digits, we consider the response correct if any of them is correct. Table 1 gives examples of several correct and incorrect model responses when the true sum is `1234`.

Models. We test a variety of current LLMs, including Claude Opus 4.1, GPT-4o, GPT-5, Gemini 2.5 Flash, Gemini 2.5 Pro, Gemini 2.5 Pro with Code Execution, and Gemma 3 27B. All models tested have have been consumer facing at some point in their life cycle.

B Error Types

Misalignment Errors More specifically, for each offset $s \in \{-10, \ldots, -1, 1, \ldots, 10\}$, we calculate $\text{offsetsum}(A, B, s) = A \cdot 10^{\max(0,s)} + B \cdot 10^{\max(0,-s)}$ and calculate the length of the shared prefix between the extracted model answer and $\text{offsetsum}(A, B, s)$. We identify the response as a misalignment error if the length is at least 6 and the extracted answer is not equal to one of the arguments.[3] Note that matching 6 digits for an offset of size at most 10 is unlikely by chance: there are at most 20 distinct 6-digit prefixes obtained from the offset sums, yet approximately 10^6 possible prefixes, so the probability of a uniformly drawn answer matching a six-digit prefix is $\approx 0.002\%$.

Table 2 shows examples of the prefixes that qualify as misalignment errors for $A = 555555$ and $B = 123456$.

[3]When the model outputs A or B, we do not consider this to be a misalignment error, even though $\text{offsetsum}(A, B, s)$ typically begins with $|s|$ digits from A or B.

Problem	Possible close carry errors
10+19 = 29	3⋆ are close carries but nothing else.
11+19 = 30	2⋆ are close carries but nothing else.
10+10 = 20	No close carries, despite 0s in answer (the rightmost column sum is 0).
199+199 = 398	No close carries, despite 9s in answer (the middle column sum is 19).

Table 3: Several addition problems and a description of the set of responses that would count as a close carry error. The character ⋆ is a place holder that can be any digit.

Model	Proportion of errors explained
claude-opus-4.1	87.9%
gpt-4o	77.5%
gpt-5	62.9%
gemini-2.5-flash	60.4%
gemini-2.5-pro	92.4%
gemma-3-27b-it	55.6%

Table 4: The fraction of all mistakes across 10,000 sum problems that are either misalignment or close carry errors.

Close Carry Errors. We use the following procedure to identify close carry errors. If the correct sum and the extracted model answer are different lengths, we pad the shorter to the left with zeros. Then we compare the extracted answer to the correct answer digit by digit from left to right—although long addition is performed from right to left, LLMs generate tokens autoregressively in the order they appear. When we reach the first digit at which an error occurs, we check if it is consistent with a close carry error:

- If the digit is too large by 1 and the column to the right has a correct column sum of 9, then we say it is a close carry error.
- If the digit is too small by 1 and the column to the right has a correct column sum of 10, then it is also a close carry error.

If the response contains no errors, or the first error does not satisfy either case above, then we do not identify it as a close carry error. Table 3 show some examples of the errors that would be considered close carries for small addition problems.

C Detailed Results

We first show the fraction of each model's errors that can be explained in Table 4.

We then look a deeper analysis of the above classification. Figure 5 shows histograms of the edit distance between the extracted model answer and the true answer for mistakes on problems of length 40 to 80. Each histogram bar is colored according to the types of the mistakes. Most models exhibit a bimodal distribution with a significant fraction of the low edit distance mistakes classified as close carry errors and many of the longer edit distance mistakes classified as misalignment errors. This is intuitive since close carry errors are local in nature, while a single misalignment can distort the entire result.

Figure 6 shows that close carry errors are significantly more prevalent than other carrying errors. The heat maps show the frequency of the model's leftmost error delta (the difference between model's digit and answer's digit) and the column sum in the next position. Close carry errors are characterized by having a delta of -1 and next column sum of 10, or a delta of +1 and a next column sum of 9. We see that these combinations are much more frequent than any other combination, although each model has its own unique behaviors as well.

Finally, for misalignment errors, Figure 7 shows histograms of the offsets producing the longest prefix match. Recall that positive offsets correspond to padding the first argument on the right with 0s. With the exception of GPT-5, the models all nearly always have a positive offset. A possible explanation is that it is common practice to write the longer argument first when adding numbers of different lengths. We see that Gemini 2.5 Flash and Pro, and Gemma 3 27B all

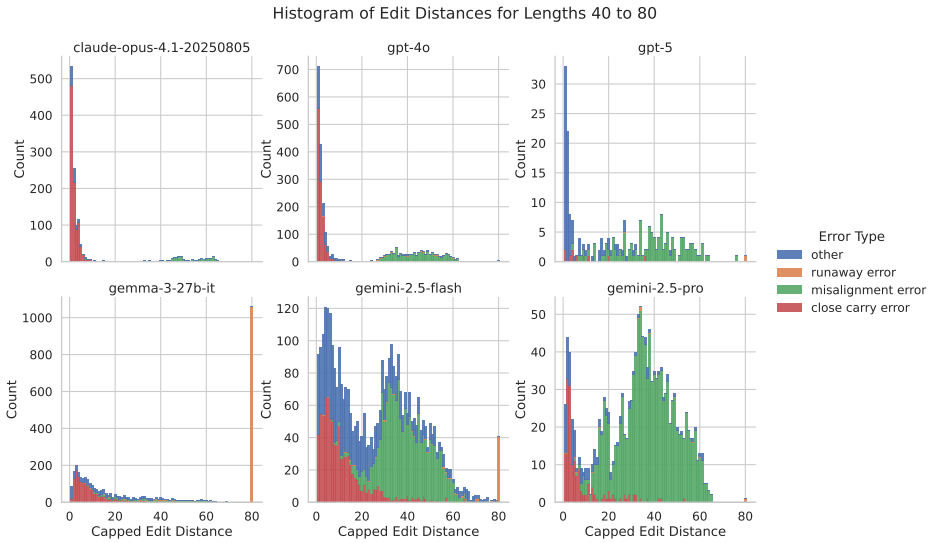

Figure 5: Distribution of error types at each edit distance.

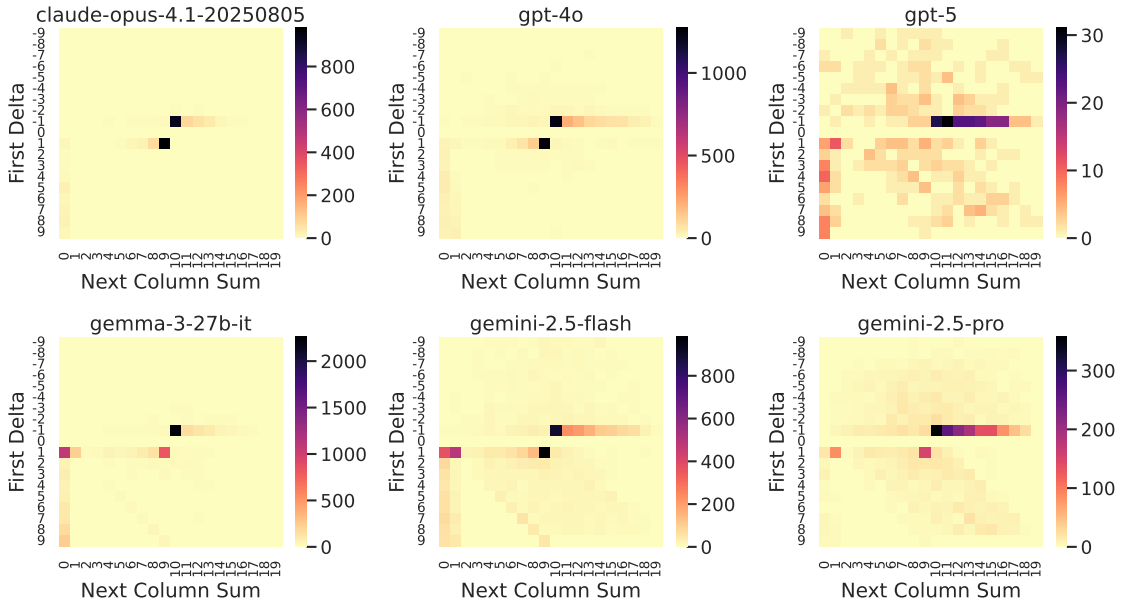

Figure 6: For each incorrect response made by a model, we find the left-most incorrect digit and calculate 1. the delta between it and the correct digit, and 2. the long addition column sum one position to the right (i.e., the column that would carry into the incorrect column). This figure shows how common each digit delta and next column sum are for each model. A significant fraction of the total count falls on the (delta=-1, next sum=10) and (delta=1, next sum = 9) positions, which exactly characterizes close carry mistakes.

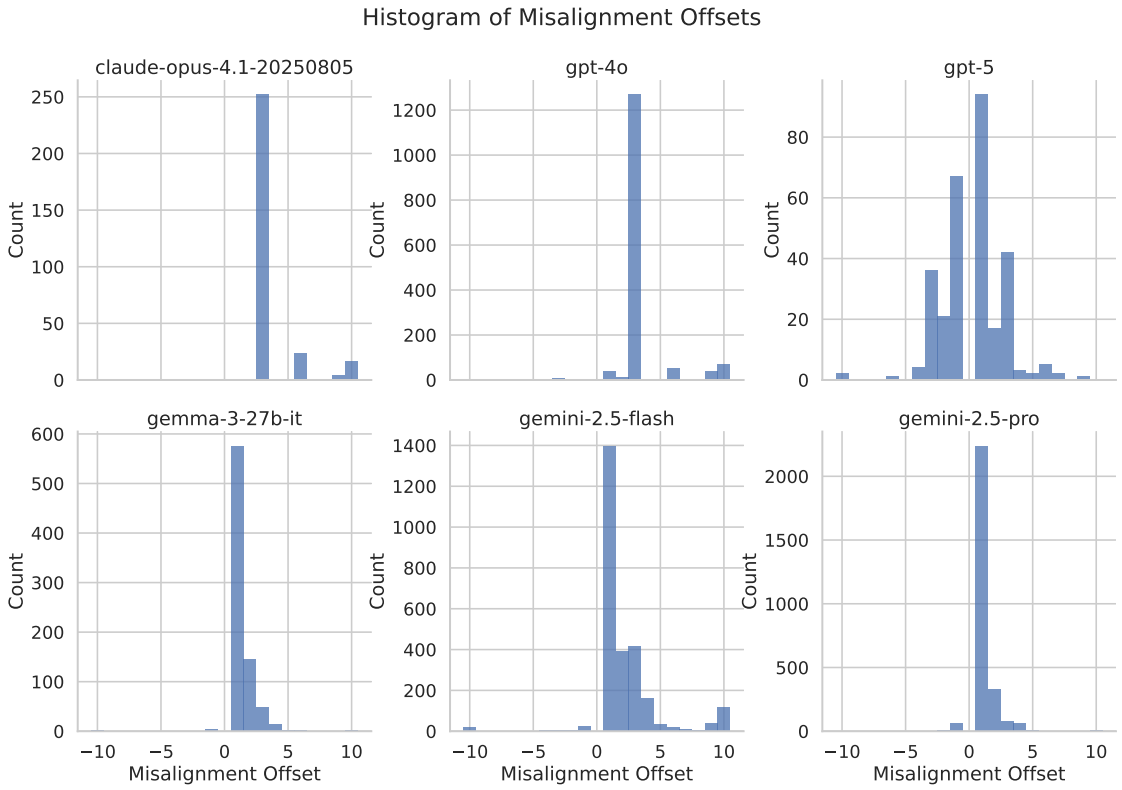

Figure 7: Argument offset that produces the longest misaligned prefix match on examples classified as a misalignment error.

typically misalign the arguments by a single digit position. On the other hand, GPT-4o and Claude Opus 4.1 favor a 3-digit misalignment. Note that these plots also show some evidence of periodicity in the offset frequencies for some models; we will return to this observation in Appendix D.1.

D Detailed Explanation of Mistakes

D.1 Tokenization and Misalignment

As observed in Figure 7, misalignment errors tend to favor certain argument offsets, which vary by model. Here we investigate further and present evidence that misalignment errors may be related to tokenization.

Figure 8 shows the unsmoothed accuracy of GPT-5 (which makes primarily misalignment errors) as a function of the argument length. Particularly for longer arguments, accuracy seems to behave in a periodic way. The right plot splits this curve into three parts, corresponding to lengths that are equivalent to 0, 1, or 2 mod 3. With the shaded regions indicating standard error, it is apparent that the accuracy of GPT-5 is significantly higher for arguments whose length is a multiple of three.

GPT-5 uses the `tiktoken` tokenizer (OpenAI, 2025b;a), in which a pre-tokenization regular expression breaks spans of digits into groups of 3 digits, followed by a group containing the remaining 1 or 2 digits if the span length is not a multiple of three. Each of these groups is then represented by a single token. As a consequence, the tokenization of a d-digit number begins with $\lfloor d/3 \rfloor$ tokens that each represent 3-digit strings, followed by a token representing a 1- or 2-digit string if $d \mod 3 \neq 0$. Figure 8 therefore suggests that GPT-5 is more likely to make mistakes precisely when 1- or 2-digit tokens are present.

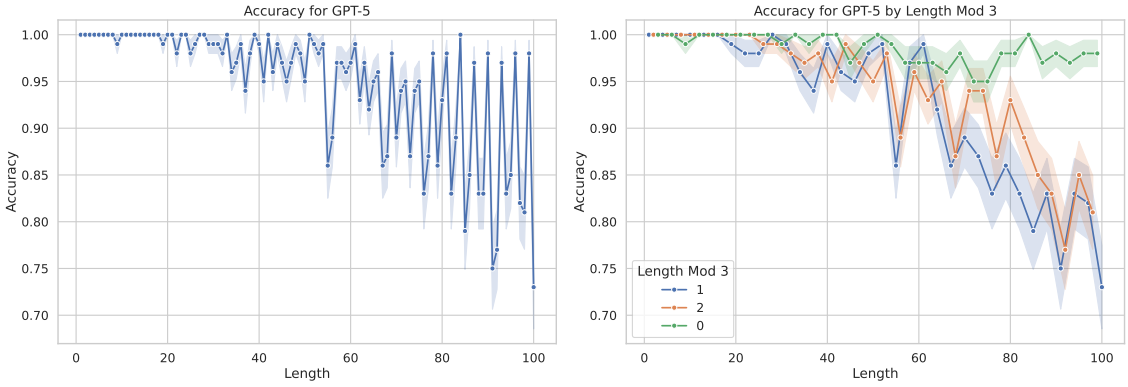

Figure 8: Left: plot showing the accuracy of GPT-5 as a function of length. Notice that every third accuracy is higher than the rest. Right: Plotting the same data but split into three curves, where each curve contains lengths that are equivalent to 0, 1, or 2 mod 3. The shaded region shows the standard error of the accuracy estimate, indicating that the separation of the 0 curve is significant.

To investigate whether this phenomenon explains misalignment errors more generally, we plot in Figure 3 the discrete Fourier transform of the misalignment error rate as a function of length for all of the models. As expected, we see a large magnitude for the frequency $1/3$ for GPT-5, corresponding to the reduced error rate on lengths divisible by 3. In addition, we see pronounced spikes at $1/3$ for GPT-4o, which also uses a 3-digit tokenizer (OpenAI, 2025a), and Claude Opus 4.1, which we find strong evidence of using a similar tokenization scheme.[4] Meanwhile, we see no spikes at $1/3$ for the other models, which we find strong evidence that they tokenize digits individually.[5]

These results suggest that misalignment errors may be exacerbated in cases where operands are encoded with tokens containing variable numbers of digits. However, we also note that models using single-digit tokenizers tended to exhibit more misalignment errors overall, so it may be that longer tokens actually help to *reduce* misalignment errors, and this effect is muted when mixed-length tokens are present.

D.2 Independence and Close Carry Errors

As shown by Figure 5, close carry errors are "local" in the sense that they tend to produce small overall edit distance between the model output and the correct answer. This suggests that a error made on one close carry may be relatively independent of other mistakes during the computation. Here we propose a simple one-parameter stochastic model of close carry errors, and show that it often matches the observed patterns of mistakes.

Assume a given addition problem involves n close carries. We imagine that the model proceeds from left to right, producing the correct output digit at each position that does not involve a close carry. Upon reaching a close carry position, the model flips a coin and makes a close carry error with probability p. If it makes an error, then the answer will be incorrect and the remainder of the output is irrelevant. If it does not make an error, it proceeds and, on reaching the next close carry position, flips another independent coin with the same bias p. Continuing in this way, the the model will ultimately produce a correct answer with probability $(1-p)^n$.

To assess whether this stochastic process is representative of the behavior of real models, we first choose a target of $n = 15$ close carries and then select only addition problems with exactly n close carries. (In general, approximately 20% of columns produce a close carry, so the selected problems have a typical length of $d = 75$.) For each model, we set p so that the predicted error matches the observed error rate on these problems, including only problems for which the model either made a close carry error or was correct. We then examine, for each selected problem, the number of *successful* close carries before the first carry error, comparing the distribution observed empirically with the one predicted by the stochastic model. As shown in Figure 4, the two distributions are often closely (though not perfectly) aligned.

[4] Specifically, when querying the `count_tokens` API for `claude-opus-4-1-20250905` (Anthropic, 2025), the increase in returned token count as a function of the number of digits d is precisely $\lceil d/3 \rceil$, matching GPT-5 and GPT-4o.

[5] See Footnote 2.

These results are consistent with the idea that some models make close carry errors in a simple, independent way, and do not correlate errors across positions. This is particularly true for Gemma 3 27B. However, not every model's behavior is fully explained in this way. In particular, several of the models show a lower propensity to make a mistake at the final close carry position, which is consistent with a general increase in accuracy for the lowest-order digits.