StreetMath: Study of LLMs' Approximation Behaviors

Anonymous Author(s)
Affiliation
Address
email

Abstract

There is a substantial body of literature examining the mathematical reasoning capabilities of large language models (LLMs), particularly their performance on precise arithmetic operations in autoregressive architectures. However, their ability to perform approximate reasoning in informal, fast-paced mathematical operations has received far less attention, especially among non-autoregressive decoder models. Our work addresses this gap by introducing StreetMath, a benchmark designed to evaluate models' approximation abilities under real-world approximation scenarios. We conduct extensive evaluations across different LLM architectures: Qwen3-4B-Instruct-2507, Qwen3-4B-Thinking-2507, Dream-v0-Instruct-7B, Falcon-Mamba-7B-Instruct, and Mamba-GPT-3B. Furthermore, we apply mechanistic interpretability techniques to probe their internal computational states. Our analysis reveals that LLMs generally attempt to compute exact values or invoke external tools even in tasks that call for approximation. Moreover, while models sometimes reach the correct answer in early layers or steps, they still consume more tokens when solving approximation tasks. Additional experiments indicate that exact and approximate arithmetic operations rely on largely separate neural components. Drawing upon research on cognitive psychology, we argue that LLMs do not exhibit cognitive miserliness in the same way humans do in street math settings. We open source our work https://anonymous.4open.science/r/StreetMath-1/

1 Introduction

Human mathematical reasoning flexibly alternates between exact calculation and rough estimation, depending on context. This adaptability—often described as "cognitive miserliness" (1)—allows people to conserve effort by using approximations when precision is unnecessary. We refer to this kind of context-driven, informal, and approximate reasoning as *street math*—the quick mental calculations people make in everyday life, such as estimating the total cost of groceries or computing a restaurant tip (e.g., leaving a 20% tip on a \$61 bill—roughly 20% of $60 \approx \$12$, which is much easier to calculate). Large language models (LLMs), in contrast, have been shown to rely on specialized internal mechanisms for arithmetic. Recent interpretability studies have uncovered Fourier-like computation circuits (2) and attention heads dedicated to mathematical processing (3). Yet it remains unclear whether these models exhibit the same context-sensitive flexibility as humans, or whether their reasoning is rigidly tied to exact solutions.

In this work, we introduce the *StreetMath* dataset, a curated collection of 1000 approximation problems drawn from everyday street math scenarios. Using this benchmark, we systematically evaluate diverse model classes, including autoregressive decoder architectures (Qwen3-4B-Instruct-2507 (4), Qwen3-4B-Thinking-2507), state-space models (Falcon-Mamba-7B (5), Mamba-GPT-3B (6)), and diffusion-based language models (Dream-v0-Instruct-7B (7)). Our experiments reveal a consistent bias across all architectures: models overwhelmingly favor exact computation, even

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

in contexts where rough estimation would suffice. Most importantly, some models achieve better approximation scores only at the cost of increased computation (tokens), which runs counter to humans' cognitive miserliness. To better understand this limitation, we examine models' rounding behavior, a fundamental operation for approximation in the street math setting. We apply linear probing to compare internal representations, finding that models' approximation on single numbers resembles human behavior: they often round numbers toward 5 or 10. In addition, models perform well at digit-level detection but struggle to generalize to word-based numbers (8).

We further investigate the neural underpinnings of these behaviors. By pruning the neurons involved in exact arithmetic (9), we uncover a surprising dynamic: removing math-specific parameters can actually improve performance on approximation tasks. This suggests that rigid, precision-oriented circuits may actively hinder flexible estimation. Additional probing into the entropy and effective ranks of intermediate layers (10) reveals similar distributions and dimensionalities between exact arithmetic operations and approximation. These findings imply that approximation does not reduce computational cost—contrary to how humans use approximation to simplify computation.

Together, these findings suggest that while LLMs have developed specialized pathways for arithmetic, they lack the human-like adaptability required for context-sensitive street math. Although LLMs are capable of approximating single numbers, they do not leverage this ability *during* the process of solving street math questions; instead, they approximate only after calculating exact answers. We conclude that LLMs do not reason about approximation questions in the same way humans do. The training corpora likely introduce this universal gap across model architectures and sizes.

2 StreetMath Dataset & Evaluations

We release 1,000 multiple-choice math reasoning problems under street math settings, covering five major topics, each with several subtopics: basket sum (sum of shopping items), discounts (buy-n-get-m-free, threshold discounts such as "$X off if you spend $Y", percentage discounts), taxes (tax before discount and tax after discount applied), units (calculating cost based on per-pound or per-kilogram prices), and tips (% on spend). Each question offers four answer options, designed to distinguish different levels of approximation capability: exact calculation, good approximation (within 20% relative error of the exact answer), mildly off (between 60% and 90% relative error), and way off (greater than 150% relative error). The benchmark not only evaluates final answers but also examines intermediate numerical evidence and the chain-of-thought (CoT) reasoning process. Any traces of exact computation or tool usage are flagged as exact math. To assess whether models exhibit cognitive miserliness, we use token count as a proxy for reasoning efficiency.

Table 1: Overall judgement counts by model with tool calls and average tokens (rounded).

Model	A	E	M	W	Uncategorized	Tool calls	Avg tokens
Qwen3-4B-Instruct-2507	445	514	40	1	0	1000	125
Qwen-4B-Thinking-2507	151	637	197	15	0	0	228
Dream-v0-Instruct-7B	0	1000	0	0	0	0	263
Falcon-Mamba-7B-Instruct	177	469	131	22	201	0	131
Mamba-GPT-3B	174	459	166	198	3	0	86

Abbreviations: A = Good approximation, E = Exact Math, M = Mildly off, W = Way off

We evaluate a range of model architectures including **autoregressive decoder**, **state-space** and **language diffusion models** across different reasoning styles (CoT vs. non-CoT) and parameter sizes (3B, 4B, 7B). The models include Qwen3-4B-Instruct-2507, Qwen3-4B-Thinking-2507, Dream-v0-Instruct-7B, Falcon-Mamba-7B-Instruct, and mamba-GPT-3B. We carefully adapt system and user prompts to each architecture to ensure fair comparisons. As shown in Table 1, LLMs across all architectures predominantly compute exact answers even when model prompt explicitly asks for approximation. When they do produce approximated answers, they typically first compute the exact value and then round it. Notably, Qwen3-4B-Thinking-2507 shows better approximation performance than Qwen3-4B-Instruct-2507, but this improvement comes at the cost of higher token usage (228 vs. 125 tokens on average) and increased deviations contrary to human cognitive miserliness. State-space models achieve similar approximation performance to Qwen3-4B-Instruct-2507 with fewer tokens but greater deviations. Dream-v0-Instruct-7B consistently produces exact answers with perfect

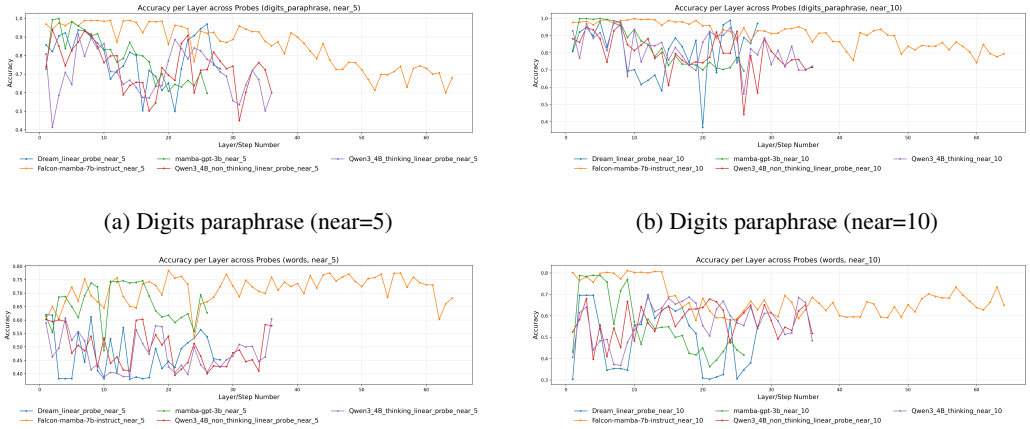
accuracy. We leave it to future work to investigate whether adjusting the steps and temperatures of Dream-v0-Instruct-7B can improve its approximation performance.

Overall, our findings indicate that LLMs tend to rely on exact arithmetic even in approximation settings, showing behavior opposite to human-like cognitive miserliness. Refer to Section B for per-topic benchmarking results.

3 Linear Probe on Rounding Behaviors

We investigate whether models encode numerical topology similar to human cognitive distance effects (11; 12) by training linear probes (13; 14) to detect nearness to multiples of 5 and 10 (15), defining proximity as exactly one integer away from the nearest multiple (e.g., 21 is near-10; 22 is not). Using simple templates to extract hidden-state representations, we evaluate five StreetMath models on digit-based ("Here is 23.") and word-based ("Consider the number twenty three.") inputs, analyzing (i) layer-wise accuracy, (ii) best-layer errors across distances 0, 1, 2+.

Digit tasks show early emergence (16) where state-space models lead: Mamba-GPT-3B reaches 99.9% and Falcon-Mamba-7B >98%, with best layers in early–middle positions (shortcut-friendly; supports early stopping), whereas Dream-v0-Instruct-7B peaks late (26th Near-5, 24th Near-10), consistent with diffusion vs. autoregressive/state-space differences. Distance-1 cases (e.g., 9, 11, 14, 16) are hardest, reflecting digit encoding (17) and calibration biases (18). Word tasks underperform across architectures, evidencing surface-form encoding and limited numerical abstraction (19; 20; 21), likely due to tokenization, pretraining bias toward digits, and separable digit/word representational clusters.



(a) Digits paraphrase (near=5)

(b) Digits paraphrase (near=10)

(c) Words (near=5)

(d) Words (near=10)

Figure 1: Accuracy per layer across model families (i) autoregressive decoder: Qwen3-4B-Thinking, Qwen3-4B-Instruct; (ii) state-space: mamba-gpt-3b, Falcon-mamba-7B-Instruct; (iii) diffusion: Dream-v0-Instruct-7B) for digits paraphrase and words tasks with near parameters 5 and 10.

4 Causal Studies

Using structured pruning to isolate parameters tied to exact arithmetic (22; 23), we find that increasing pruning does not necessarily hurt StreetMath performance: aside from Qwen3-4B-Instruct-2507, most models remain stable or even improve under moderate pruning, contradicting the intuition that reduced capacity uniformly impairs numerical reasoning. Pruning effects diverge by benchmark: MMLU and RACE are similarly resilient, whereas GSM8K is extremely sensitive—even slight pruning collapses accuracy to near zero across all models—implicating a specialized, fragile neuron subset for exact arithmetic while StreetMath and language-heavy tasks rely on more distributed representations. These patterns align with prior results (22), suggesting a dual pathway: (i) localized, brittle circuits for exact arithmetic that fail under pruning, and (ii) distributed, robust circuits for approximation and text-heavy reasoning, where moderate pruning can denoise and improve performance—consistent

with StreetMath being tackled more as context-driven linguistic estimation than strict mathematical computation.

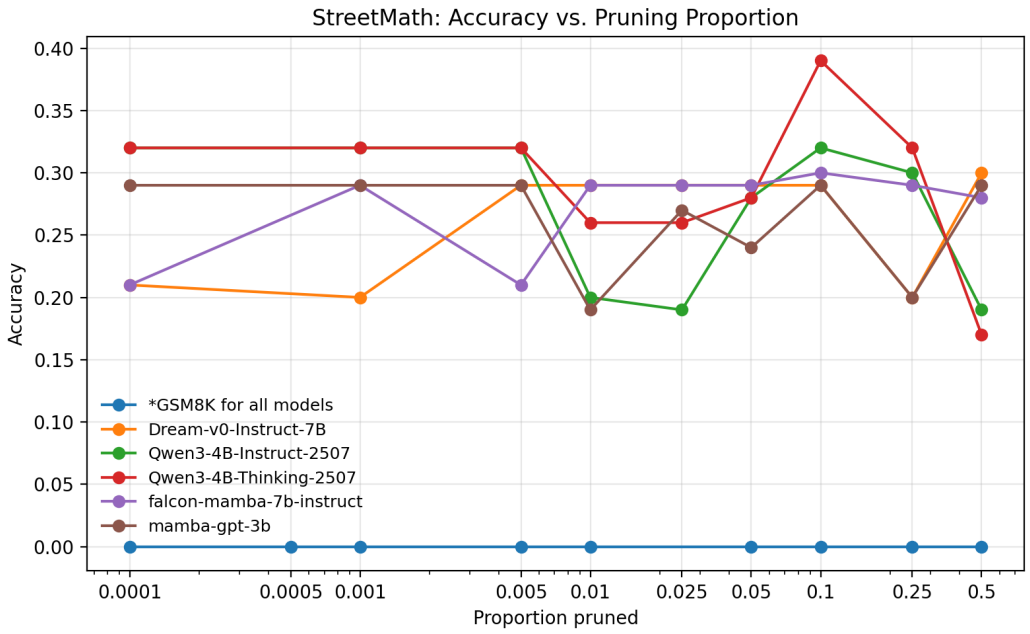


Figure 2: Effect of structured pruning on task performance for all models. Accuracy is plotted against the proportion of parameters pruned for StreetMath and GSM8K benchmarks

5 Layer-wise Studies

The layer-wise analyses (10) reveal a broadly U-shaped evolution of spectral entropy and effective rank (high at input, dipping early, then rising) across models and tasks, with Falcon-Mamba-7B on StreetMath as the main exception. GSM8K runs of Qwen3-4B-Instruct-2507 show a pronounced dip by the first third of layers and a steady increase. Notably, both GSM8K and StreetMath runs exhibit elbow-like transitions at comparable depths, consistent with early compression and later re-expansion seen in shortcut reasoning (24). This observation supports the view that approximation in StreetMath does not help models reach solutions more efficiently, showing the opposite of human cognitive miserliness (25).

It is evident from our experiments that task-specific effects emerge across the models. StreetMath runs typically show higher late-layer entropy and effective rank than GSM8K for the same model, along with larger transition distances. This pattern indicates not only higher variability across models but also more sustained representational expansion and stronger late-stage adjustments. By contrast, GSM8K often consolidates into a stable mid-layer corridor with very high cosine similarity and minimal angular changes. These observations support our causal study results that models use a more diverse set of neurons when handling street math-type questions while dedicating to a small set of neurons when handling exact arithmetic operations. For details, refer to E.

6 Conclusion

We curated the *StreetMath* benchmark to reveal LLMs' lack of cognitive miserliness in street math settings. Although these models possess single-number rounding capability, they do not leverage it to reduce computational effort. We further discovered that models use a more diverse set of neurons when handling street-math-style questions while dedicating a small set of neurons to exact arithmetic operations.

References

[1] D. Kahneman, *Thinking, fast and slow*. Farrar, Straus and Giroux, 2011.

[2] T. Zhou, D. Fu, V. Sharan, and R. Jia, "Pre-trained large language models use fourier features to compute addition," *arXiv preprint arXiv:2406.03445*, 2024.

[3] Z. Yu and S. Ananiadou, "Interpreting arithmetic mechanism in large language models through comparative neuron analysis," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3293–3306, 2024.

[4] Q. Team, "Qwen3 technical report," 2025.

[5] J. Zuo, M. Velikanov, D. E. Rhaiem, I. Chahed, Y. Belkada, G. Kunsch, and H. Hacid, "Falcon mamba: The first competitive attention-free 7b language model," 2024.

[6] CobraMamba, "Mamba-gpt-3b." `https://huggingface.co/CobraMamba/mamba-gpt-3b`, 2023. Hugging Face model card; Apache-2.0 license.

[7] J. Ye, Z. Xie, L. Zheng, J. Gao, Z. Wu, X. Jiang, Z. Li, and L. Kong, "Dream 7b: Diffusion large language models," *arXiv preprint arXiv:2508.15487*, 2025.

[8] O. Levy and M. Geva, "Language models encode numbers," *arXiv preprint*, 2024.

[9] B. R. Christ, Z. Gottesman, J. Kropko, and T. Hartvigsen, "Math neurosurgery: Isolating language models' math reasoning abilities using only forward passes," *arXiv preprint*, 2025.

[10] O. Skean, M. R. Arefin, D. Zhao, N. Patel, J. Naghiyev, Y. LeCun, and R. Shwartz-Ziv, "Layer by layer: Uncovering hidden representations in language models." version: 2.

[11] S. Dehaene, *The number sense: How the mind creates mathematics*. OUP USA, 2011.

[12] R. S. Moyer and T. K. Landauer, "Time required for judgements of numerical inequality," *Nature*, vol. 215, no. 5109, pp. 1519–1520, 1967.

[13] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," *arXiv preprint arXiv:1610.01644*, 2016.

[14] J. Hewitt and C. D. Manning, "A structural probe for finding syntax in word representations," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019.

[15] J. De Brauwer, T. Verguts, and W. Fias, "The representation of multiplication facts: Developmental changes in the problem size, five, and tie effects," *Journal of Experimental Child Psychology*, vol. 94, no. 1, pp. 43–66, 2006.

[16] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469, IEEE, 2016.

[17] A. A. Levy and M. 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)*, pp. 385–395, 2025.

[18] C. Lovering, M. Krumdick, V. D. Lai, S. Ebner, N. Kumar, V. Reddy, R. Koncel-Kedziorski, and C. Tanner, "Language model probabilities are not calibrated in numeric contexts," *arXiv preprint arXiv:2410.16007*, 2024.

[19] R. T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," *arXiv preprint arXiv:1902.01007*, 2019.

[20] Y. Belinkov and J. Glass, "Analysis methods in neural language processing: A survey," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 49–72, 2019.

[21] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, 2016.

[22] B. R. Christ, Z. Gottesman, J. Kropko, and T. Hartvigsen, "Math neurosurgery: Isolating language models' math reasoning abilities using only forward passes."

[23] D. Rai, Y. Zhou, S. Feng, A. Saparov, and Z. Yao, "A practical review of mechanistic interpretability for transformer-based language models."

[24] M. Ding, H. Liu, Z. Fu, J. Song, W. Xie, and Y. Zhang, "Break the chain: Large language models can be shortcut reasoners."

[25] D. L. Jiang, S. Ye, L. Zhao, and B. Gu, "Do reductions in search costs for partial information on online platforms lead to better consumer decisions? evidence of cognitive miser behavior from a natural experiment," p. isre.2022.0432.

[26] O. Roy and M. Vetterli, "The effective rank: A measure of effective dimensionality," in *2007 15th European Signal Processing Conference*, pp. 606–610, IEEE, 2007.

[27] G. Srivastava, A. Hussain, S. Srinivasan, and X. Wang, "LMThinkBench: Towards basic math reasoning and overthinking in large language models."

[28] K. Paster, M. D. Santos, Z. Azerbayev, and J. Ba, "Openwebmath: An open dataset of high-quality mathematical web text," *arXiv preprint arXiv:2310.06786*, 2023.

[29] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, *et al.*, "Solving quantitative reasoning problems with language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3843–3857, 2022.

[30] I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar, "GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models." Apple.

[31] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. K. Li, Y. Gong, Z. Jin, X. Wang, *et al.*, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv preprint arXiv:2402.03300*, 2024.

[32] Y. Ding *et al.*, "Break the chain: Large language models with heuristics," *arXiv preprint*, 2024.

[33] W. Zhao, J. Guo, Y. Deng, X. Sui, Y. Hu, Y. Zhao, W. Che, B. Qin, T.-S. Chua, and T. Liu, "Exploring and exploiting the inherent efficiency within large reasoning models for self-guided efficiency enhancement."

[34] M. Skean *et al.*, "Layer by layer: Uncovering mathematical reasoning," *arXiv preprint*, 2025.

[35] X. Sun *et al.*, "Probing for arithmetic errors in language models," *arXiv preprint*, 2025.

[36] A. Saynova *et al.*, "Fact recall, heuristics or pure computation," *arXiv preprint*, 2025.

[37] S. Kantamneni and M. Tegmark, "Language models use trigonometric functions," *arXiv preprint*, 2025.

[38] W. Zhu *et al.*, "Language models encode the concept of numeric magnitude," *arXiv preprint*, 2025.

[39] R. Shah *et al.*, "Numeric magnitude comparison," *arXiv preprint*, 2023.

[40] J. Li *et al.*, "Diffusion language models," *arXiv preprint*, 2025.

[41] S. Jelassi, D. Brandfonbrener, S. M. Kakade, and E. Malach, "Repeat after me: Transformers are better than state space models at copying," in *International Conference on Machine Learning*, pp. 21502–21521, 2024.

[42] T. Schick, J. Dwivedi-Yu, R. Dessà, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[43] D. Das, D. Banerjee, S. Manocha, and A. Baral, "Mathsensei: A tool-augmented large language model for mathematical reasoning," *arXiv preprint arXiv:2402.17231*, 2024.

[44] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "Pal: Program-aided language models," *International Conference on Machine Learning*, 2023.

[45] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *arXiv preprint arXiv:2211.12588*, 2022.

[46] M. Dietz and D. Klakow, "Igc: Integrating a gated calculator," *arXiv preprint*, 2025.

[47] K. Lauter *et al.*, "Machine learning for modular arithmetic," *arXiv preprint*, 2024.

[48] A. Nikankin *et al.*, "Arithmetic without algorithms," *arXiv preprint*, 2025.

[49] L. Gambardella *et al.*, "Language models do hard arithmetic," *arXiv preprint*, 2024.

[50] C. Lovering *et al.*, "Language model probabilities," *arXiv preprint*, 2024.

[51] J. Ahn, R. Verma, R. Lou, D. Liu, R. Zhang, and W. Yin, "Large language models for mathematical reasoning: Progresses and challenges."

A Experiment Setup

A.1 Model Selection

To examine how different architectures perform under the street math setting, we selected representative models from autoregressive transformer, diffusion-based LLM, and state-space families. Given computational constraints, we restricted our study to small- and medium-sized models. To ensure reproducibility and enable deeper investigation of internal mechanisms, we further limited our selection to open-source models with publicly available weights. Because the task requires models to follow prompts reliably and generate multiple-choice responses, we focused on instruction-tuned and thinking models. Within these constraints, we also sought to preserve meaningful comparisons, such as chain-of-thought versus instruction-only models, as well as cross-architecture and cross-size contrasts.

Accordingly, our study evaluates Qwen3-4B-Instruct-2507, Qwen3-4B-Thinking-2507, Dream-v0-Instruct-7B, Falcon-Mamba-7B, and Mamba-GPT-3B. All models are initialized with the default parameters.

A.2 Hardware specifications

We conducted all experiments on a single NVIDIA A10 GPU hosted on RunPod, using an Ubuntu 22.04 operating system with CUDA version 12.8.1.

B StreetMath dataset and benchmark result

B.1 Data Curation

StreetMath targets everyday "street math," emphasizing fast estimation over exact arithmetic. It contains multiple-choice questions across shopping and daily-life contexts: basket totals, discounts (percentage-off, BOGO, buy-n-get-m, threshold coupons), taxes (pre/post-discount), unit conversions (lb-oz, kg-g), and tips. Prompts explicitly nudge for approximate reasoning ("about how much") to elicit human-style rounding.

Each question has four options: the exact value; a "good approximation" within 20% relative error (correct); a "mildly off" option; and a "way off" option (fractional or multi-fold). Choices are shuffled

Model	Topic	Good approx	Exact math	Mildly off	Way off	Uncategorized	N
Qwen3-4B-Instruct-2507	basket_sum	86	154	1	0	0	241
	discounts	15	220	7	0	0	242
	taxes	40	132	1	0	0	173
	units	22	150	0	0	0	172
	tips	22	150	0	0	0	172
Qwen-4B-Thinking-2507	basket_sum	46	104	55	36	0	241
	discounts	80	61	51	50	0	242
	taxes	40	45	46	42	0	173
	units	35	84	22	31	0	172
	tips	28	68	40	36	0	172
Dream-v0-Instruct-7B	basket_sum	0	241	0	0	0	241
	discounts	0	242	0	0	0	242
	taxes	0	173	0	0	0	173
	units	0	172	0	0	0	172
	tips	0	172	0	0	0	172
Falcon-Mamba-7B	basket_sum	47	106	43	0	45	241
	discounts	50	108	61	5	18	242
	taxes	38	63	47	0	25	173
	units	8	94	7	14	49	172
	tips	11	77	4	0	80	172
Mamba-GPT-3B	basket_sum	51	97	46	47	0	241
	discounts	43	111	35	53	0	242
	taxes	29	59	39	43	3	173
	units	32	78	31	31	0	172
	tips	19	114	15	24	0	172

Table 2: Benchmark results: Counts by topic for all models.

A–D, with metadata storing numeric values. Spacing ensures clear separation: mild $\geq 60\%$ and way $\geq 150\%$.

Good approximations follow deterministic rounding rules. Basket totals round prices to dollars, then sum and drop cents. Discounts round prices to dollars, rates to nearest 5%, pair BOGO (buy one get one) items by price, and compute buy-n-get-m deterministically. Threshold coupons apply to a rounded subtotal. Taxes round bases and rates (5% steps) before dropping cents. Unit costs round prices and weights. Tips apply percentages to subtotals rounded to \$5/\$10 buckets.

Data generation is deterministic given a seed. Templates randomize prices, quantities, and rates. Outputs are JSONL lines with `id`, `topic`, `prompt`, `choices`, `labels`, `correct_label`, and `metadata` (exact, good, mild, way). Splits are controllable by topic weights. A validator enforces spacing and alignment.

B.2 StreetMath Benchmark

The benchmark evaluates LLMs on StreetMath via local JSONL or hosted dataset (`LuxMuseAI/StreetMathDataset`). The system prompt encourages estimation and discourages exact calculation. Models must output: "Final choice: `<A|B|C|D>`", "Answer: `<numeric>`", and "Reasoning: `<short sentence>`"; optional inner thoughts appear in `<think>...</think>`. The runner supports OpenAI-compatible APIs, local Transformers, and Ollama.

Outputs are parsed for choice, numeric answer, reasoning, and optional tool calls. If only a number is given, the closest choice is inferred. Labels: exact = "Exact math," good = "Good approximation," mild/way = "Mildly off"/"Way off." We use the count of Good approximation as evaluation metrics to avoid giving arbitrary weights to each choice.

Each sample yields a JSON record with prompt, predictions, reasoning, token/latency, and judgement. A summary aggregates mean scores, label counts, accuracy by topic, tool-call frequency, and average resource use. This setup cleanly separates approximation skill from exact computation preference while ensuring reproducibility across models and backends.

C Linear Probe

C.1 Experimental Setup

Task Definition: We train linear probes to detect numerical proximity concepts, specifically whether numbers are "near" multiples of 5 or 10. For near-5 detection, proximity is defined as $\min(|n \bmod 10 - 0|, |n \bmod 10 - 5|, |n \bmod 10 - 10|) \leq 1$, covering digits {0, 1, 4, 5, 6, 9}. For near-10 detection, proximity is defined as $\min(|n \bmod 10 - 0|, |n \bmod 10 - 10|) \leq 1$, covering digits {0, 1, 9}.

Data Generation: We generated 4,000 training samples and 1,500 validation samples per condition. Numbers were randomly sampled from [0, 9999] and embedded into descriptive templates. Two template sets were used:

- Template A: “Consider the number {n}.”, “Let x = {n}.”, “Value: {n}”, etc.
- Template B: “Here is {n}.”, “We study the scalar {n}.”, “Write down {n} and continue.”, etc.

Numbers were presented in two surface forms: digits (“25”) and words (“twenty five”) using the num2words library with normalization (hyphens and commas removed, lowercase).

Training Protocol: We used a two-stage streaming approach to handle memory constraints:

1. **Standardization:** StandardScaler fitted per layer using partial_fit() with mean centering disabled
2. **Classification:** SGD logistic regression with optimal learning rate, L2 regularization ($\alpha = 10^{-4}$), and single-epoch updates

C.2 Evaluation Methodology

Cross-Template Validation: Three validation sets tested different robustness aspects: 1.Training: Template A + digits; 2. Validation A: Template B + digits (template robustness); 3. Validation W: Template A + words (cross-modal transfer).

Error Analysis: We analyzed error patterns at the best-performing layer (highest accuracy) across distance buckets. For near-5: distances 0, 1, 2+ . For near-10: distances 0-5 maintained separately. We also examined errors by rounding direction: -1 (round down closer), 0 (exact multiple), +1 (round up closer).

Layer Selection Rationale: We analyzed the best-performing layer rather than layer averages because: (1) it reveals models’ optimal proximity detection capabilities, (2) it avoids noise from suboptimal layers that could mask genuine patterns, (3) it aligns with interpretability goals of understanding whether models *can* learn proximity concepts.

Layer Sampling: We probed every layer (stride=1) for comprehensive analysis, skipping only embedding layers (layer 0).

Statistical Measures: Accuracy per layer, error rates by distance/direction, best layer identification. Results averaged over single runs with fixed random seeds (1337) for reproducibility.

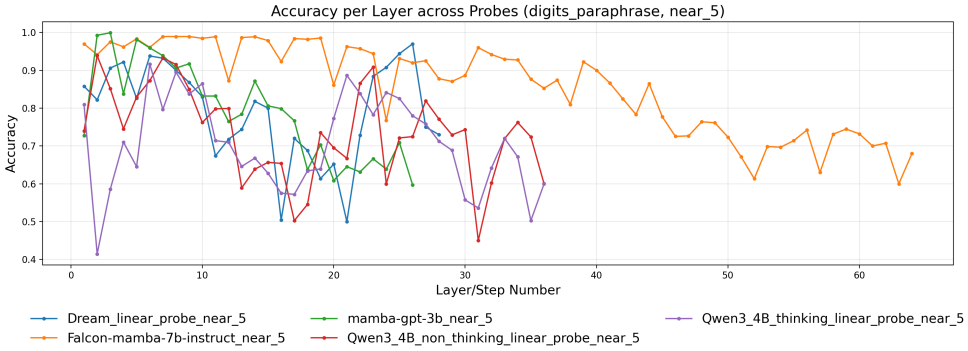


Figure 3: Accuracy per layer for digits paraphrase with near parameter 5 across autoregressive decoder based models including Qwen3-4B-Thinking, Qwen3-4B-Non-Thinking, state-space based model including mamba-gpt-3b, Falcon-mamba-7B-Instruct, and diffusion based model Dream-v0-Instruct-7B.

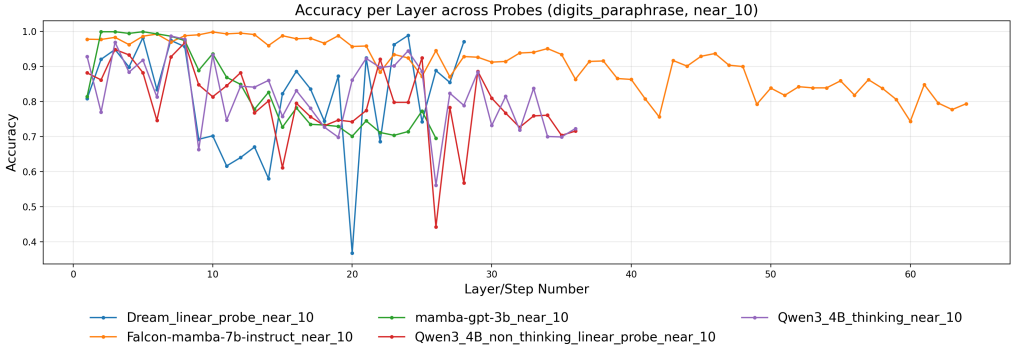


Figure 4: Accuracy per layer for digits paraphrase with near parameter 10 across autoregressive decoder based models including Qwen3-4B-Thinking, Qwen3-4B-Non-Thinking, state-space based model including mamba-gpt-3b, Falcon-mamba-7B-Instruct, and diffusion based model Dream-v0-Instruct-7B.

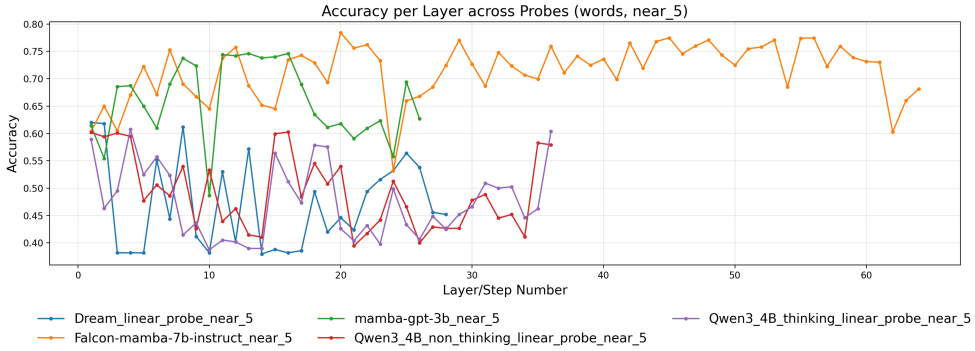


Figure 5: Accuracy per layer for words with near parameter 5 across autoregressive decoder based models including Qwen3-4B-Thinking, Qwen3-4B-Non-Thinking, state-space based model including mamba-gpt-3b, Falcon-mamba-7B-Instruct, and diffusion based model Dream-v0-Instruct-7B.

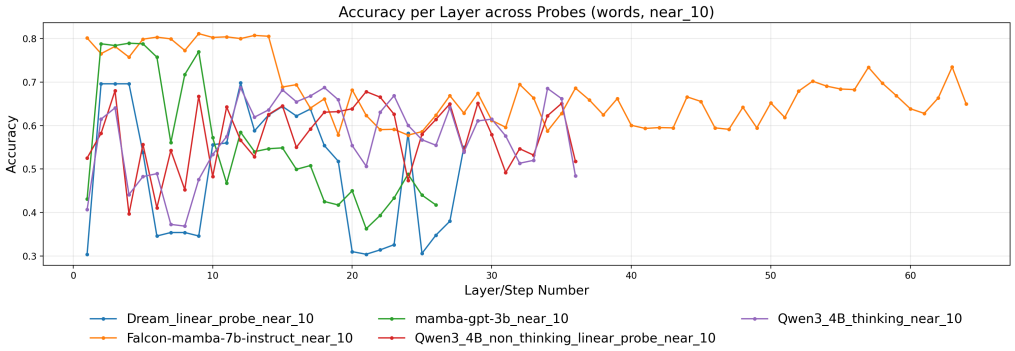


Figure 6: Accuracy per layer for words with near parameter 10 across autoregressive decoder based models including Qwen3-4B-Thinking, Qwen3-4B-Non-Thinking, state-space based model including mamba-gpt-3b, Falcon-mamba-7B-Instruct, and diffusion based model Dream-v0-Instruct-7B.

Table 3: Comprehensive Near-5 Digit Analysis: Performance and Error Patterns at the best layer. Acc = Accuracy; Err = Error rate

Model	**Peak Acc**	**Best Layer**	**Err (0)**	**Err (1)**	**Err (2)**
Qwen3-4B-Instruct	0.939	2	0.4%	5.5%	9.4%
Qwen3-4B-Thinking	0.917	6	7.2%	14.6%	2.5%
Dream-7B	0.970	26	4.2%	4.8%	0.5%
Falcon-Mamba-7B-Instruct	0.989	7	0.7%	0.6%	1.7%
Mamba-GPT-3B	0.999	3	0.4%	0.0%	0.0%

Table 4: Comprehensive Near-5 (Words) Analysis: Performance and Error Patterns at the best layer. Acc = Accuracy; Err = Error rate

Model	**Peak Acc**	**Best Layer**	**Err (0)**	**Err (1)**	**Err (2)**
Qwen3-4B-Instruct	0.603	16	7.0%	4.0%	94.3%
Qwen3-4B-Thinking	0.607	4	0.4%	0.6%	100.0%
Dream-7B	0.620	1	0.0%	0.0%	99.5%
Falcon-Mamba-7B-Instruct	0.784	20	4.2%	2.7%	50.5%
Mamba-GPT-3B	0.746	13	2.1%	0.0%	64.2%

Table 5: Comprehensive Near-10 Analysis: Performance and Error Patterns at the Best Layer

Model	**Peak Acc**	**Best Layer**	**Err (0)**	**Err (1)**	**Err (2)**	**Err (3)**	**Err (4+)**
Qwen3-4B-Instruct	0.967	8	4%	12%	1%	1%	0%
Qwen3-4B-Thinking	0.987	7	1%	3%	3%	0%	1%
Dream-7B	0.988	24	2%	5%	0%	0%	0%
Falcon-Mamba-7B-Instruct	0.998	10	1%	0%	1%	0%	0%
Mamba-GPT-3B	0.999	2	0%	0%	0%	0%	0%

Table 6: Comprehensive Near-10 (Words) Analysis: Performance and Error Patterns at the Best Layer

Model	**Peak Acc**	**Best Layer**	**Err (0)**	**Err (1)**	**Err (2)**	**Err (3)**	**Err (4+)**
Qwen3-4B-Instruct	0.680	3	96%	98%	3%	4%	3%
Qwen3-4B-Thinking	0.687	18	97%	96%	4%	2%	2%
Dream-7B	0.698	12	98%	100%	0%	0%	0%
Falcon-Mamba-7B-Instruct	0.811	9	67%	58%	0%	0%	0%
Mamba-GPT-3B	0.789	4	74%	57%	2%	5%	2%

D Causal Study

We adapt the **MathNeuro** codebase(22) to study pruning and scaling in instruction-tuned LMs. For each calibration corpus (a CSV with *instruction* and *response* columns), we estimate parameter importance by registering forward hooks on all `Linear` layers and accumulating mean activation magnitudes weighted by the corresponding weight magnitudes over 200 calibration samples. We then construct a keep-mask that retains the top $p\%$ of parameters, where $p \in \{0.01\%, 0.1\%, 0.5\%, 1\%, 2.5\%, 5\%, 10\%, 25\%, 50\%\}$.

Due to compute constraints, each setting is run once using bootstrap samples (≤ 500 examples) drawn from both the training set (CSV with *question*, *solution*, and *answer* fields) and each calibration set. For every pruning proportion, we reload the model (`AutoModelForCausalLM`, `bfloat16`, `device_map=auto`; Dream models are wrapped for `lm_eval` compatibility), apply the mask, and evaluate performance using the **EleutherAI LM Evaluation Harness** on user-specified tasks.

To manage compute, per-task evaluation is capped at 1,000 items, and prompts are truncated to 256 tokens. When no `lm_eval` tasks are provided, a lightweight multiple-choice evaluator is used. For **GSM8K**, evaluation is limited to 1,000 samples. For **StreetMath**-style multiple choice, we treat a "good approximation" judgment as correct.

All results are saved per model, per task and per pruning proportion in the specified results directory.

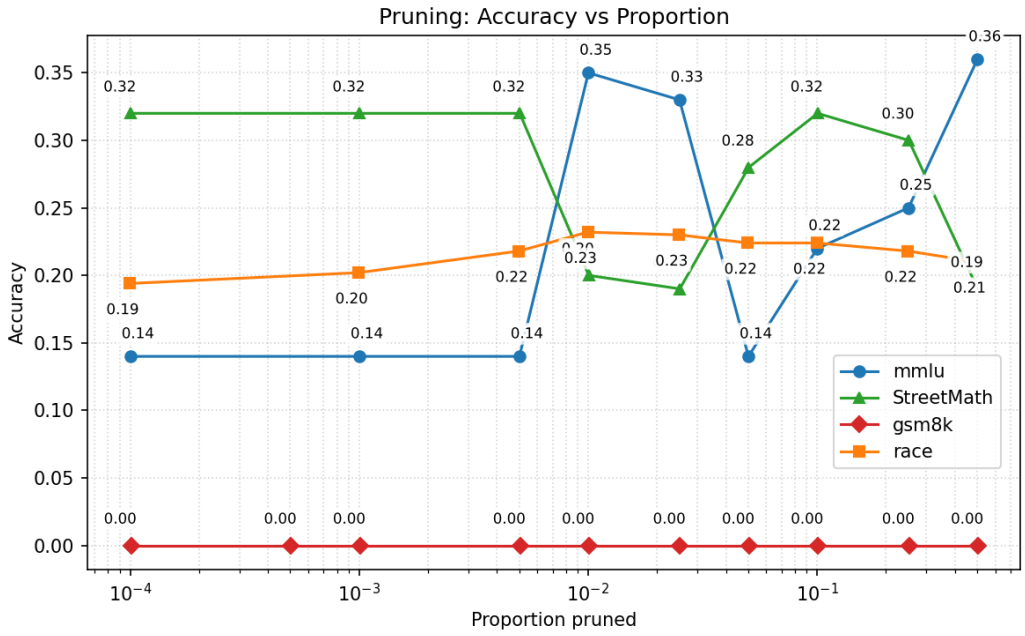


Figure 7: Effect of structured pruning on task performance for Qwen3-4B-Instruct-2507. Accuracy is plotted against the proportion of parameters pruned for four benchmarks (MMLU, StreetMath, GSM8K, and RACE).

E Layerwise Study

The experiments implement a two-stage pipeline that first extracts layerwise diagnostics from transformer models on mathematical reasoning corpora and then aggregates and visualizes these diagnostics across many prompts.

In the first stage, model-specific analysis scripts (for example, Dream-v0-Instruct-7B, Qwen3-4B variants, Mamba-GPT-3B, and Falcon-mamba-7B-Instruct) load a Hugging Face model and tokenizer and evaluate it on a chosen dataset split. The workflows support both the GSM8K test split and a StreetMath test set. For each prompt, the scripts request hidden states, and compute a suite of metrics for every layer. Intra-layer measurements include spectral entropy and effective rank (26)

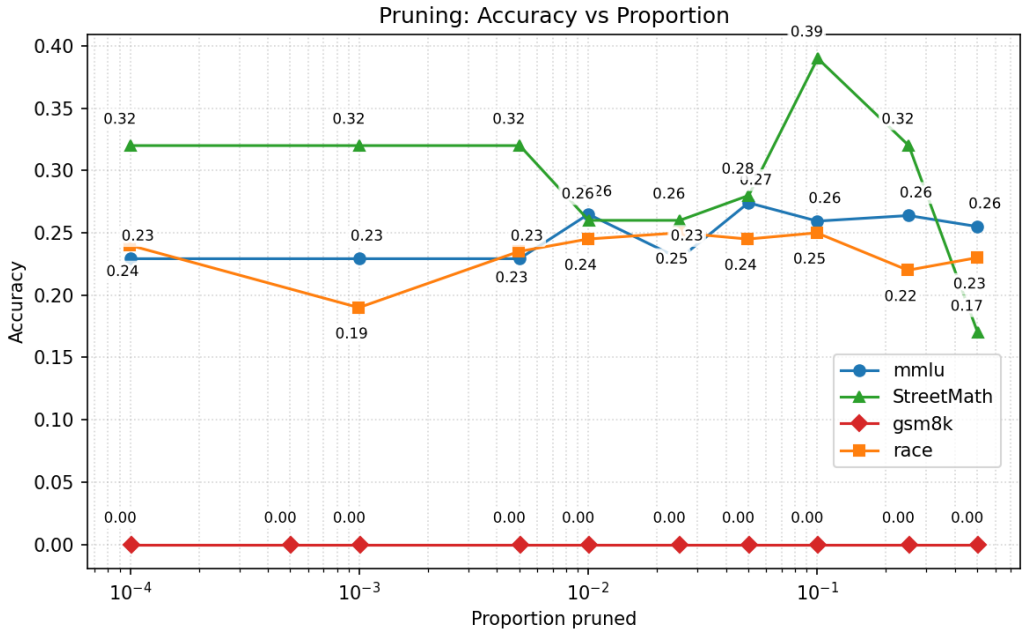


Figure 8: Effect of structured pruning on task performance for Qwen3-4B-Thinking-2507. Accuracy is plotted against the proportion of parameters pruned for four benchmarks (MMLU, StreetMath, GSM8K, and RACE).

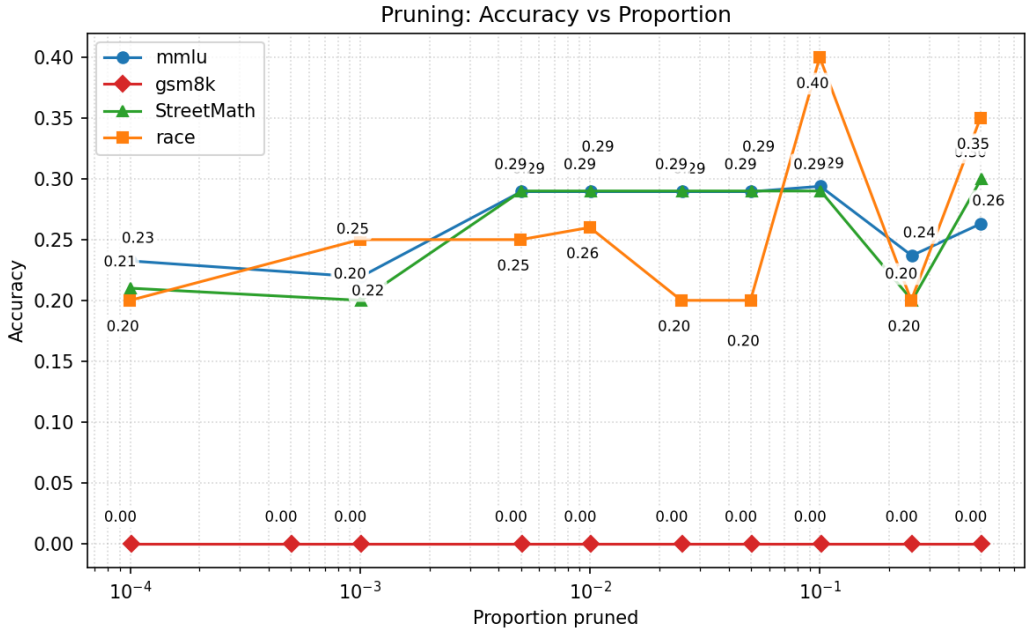


Figure 9: Effect of structured pruning on task performance for Dream-v0-Instruct-7B. Accuracy is plotted against the proportion of parameters pruned for four benchmarks (MMLU, StreetMath, GSM8K, and RACE).

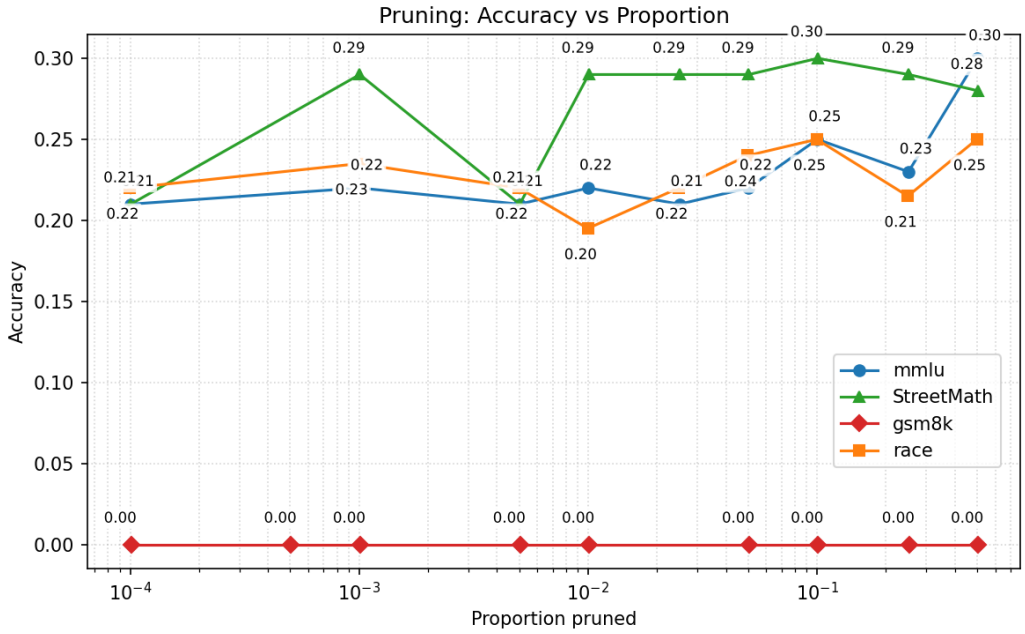


Figure 10: Effect of structured pruning on task performance for Falcon-Mamba-7B-Instruct. Accuracy is plotted against the proportion of parameters pruned for four benchmarks (MMLU, StreetMath, GSM8K, and RACE).

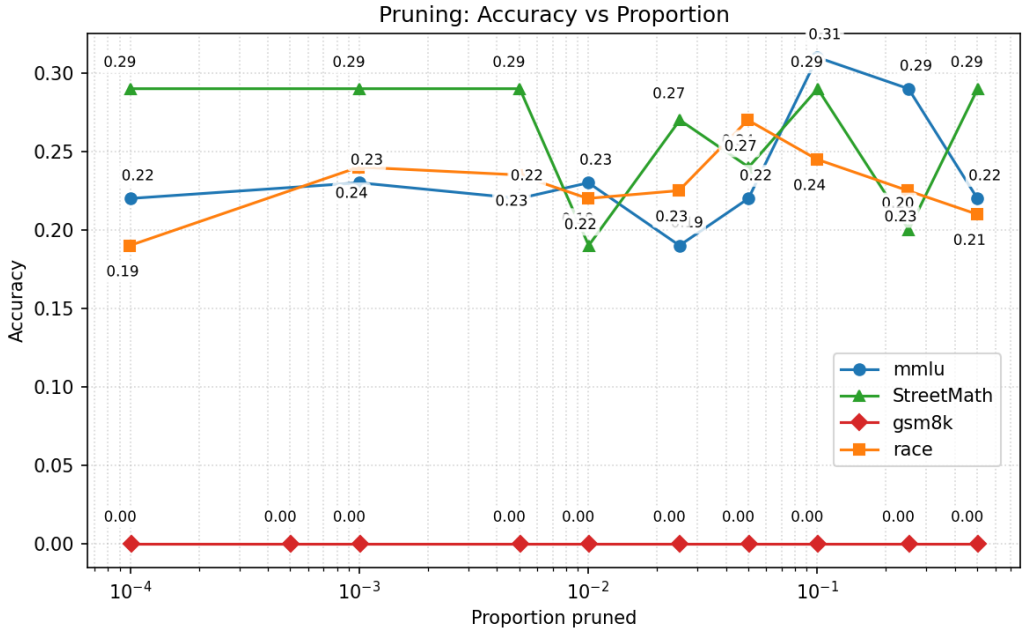


Figure 11: Effect of structured pruning on task performance for Mamba-GPT-3B. Accuracy is plotted against the proportion of parameters pruned for four benchmarks (MMLU, StreetMath, GSM8K, and RACE).

obtained from singular-value spectra, activation entropy computed from histogram estimates, the trace of the covariance matrix as a proxy for Gaussian complexity, gradient norms approximated by the variance of hidden activations, logit-lens proxy scores, and attention entropy when attention weights are present. Inter-layer measurements quantify how the representation changes from one layer to the next through cosine similarity, L2 distance, and angular distance. Each prompt therefore contributes a record containing these per-layer vectors, along with metadata, to a JSON file. Due to computational constraint, we limit each dataset to 1000 samples.

The second stage consolidates these per-prompt records. The script reads a results JSON and computes the sample mean and the sample standard deviation across prompts for every metric and for every layer index. Because the raw results may mix series of slightly different lengths, the aggregation is performed at the most common length observed for each metric, ensuring that elementwise statistics are well-defined and not dominated by outliers in shape.

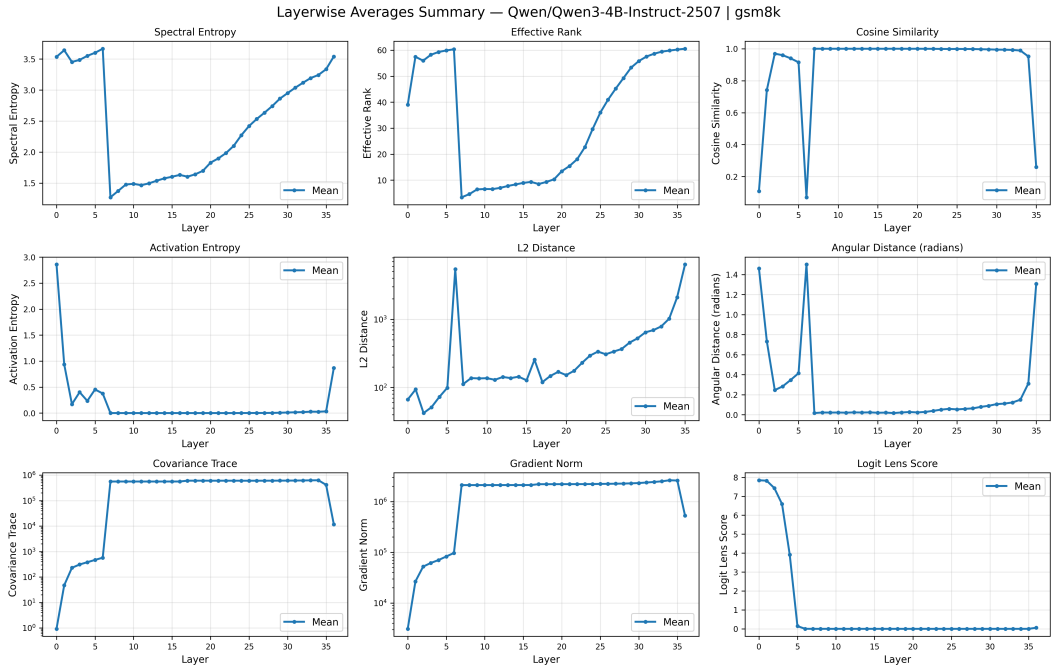


Figure 12: Layerwise Average Summary - Qwen3-4B-Instruct-2507 on GSM8K

F Related Work

F.1 The Approximation Gap in Mathematical Reasoning

Current mathematical reasoning research exhibits a systematic bias toward exact computation, creating a fundamental blind spot in our understanding of numerical intelligence. Zhou et al. (2) demonstrated that LLMs use specialized Fourier mechanisms for precise arithmetic, while Yu and Ananiadou (3) identified localized attention heads for exact operations. Kahneman (1)—adaptively reduces computational effort when an approximation suffices. These findings systematically overlook cognitive flexibility, instead celebrating models that can perform precise calculations while ignoring whether they can engage in the contextually appropriate approximation that characterizes genuine mathematical understanding. These mechanistic insights, while valuable, represent a narrow conception of mathematical reasoning that prioritizes precision over cognitive flexibility. Recent work by Srivastava et al. on LMThinkBench (27) reveals that models achieve high accuracy but at the cost of unnecessarily complex reasoning paths; a pattern consistent with systems that lack the cognitive control mechanisms necessary for adaptive approximation. When models cannot modulate their computational precision based on contextual demands, they default to maximum effort regardless of whether such precision is warranted or efficient. Highlighting the gap between computational capability and efficient reasoning.

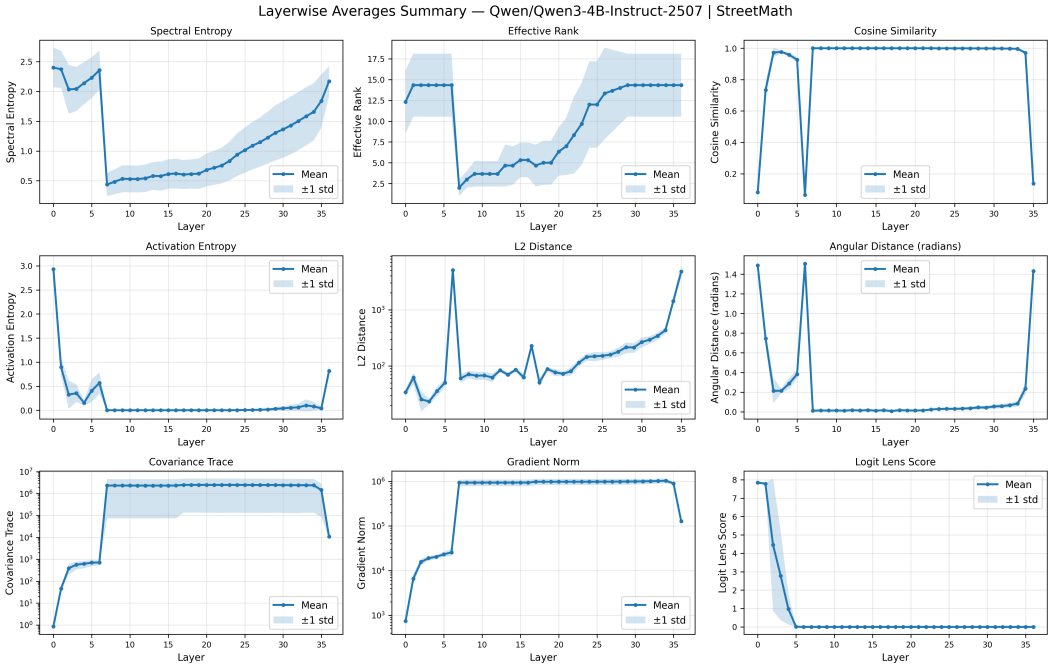


Figure 13: Layerwise Average Summary - Qwen3-4B-Instruct-2507 on StreetMath

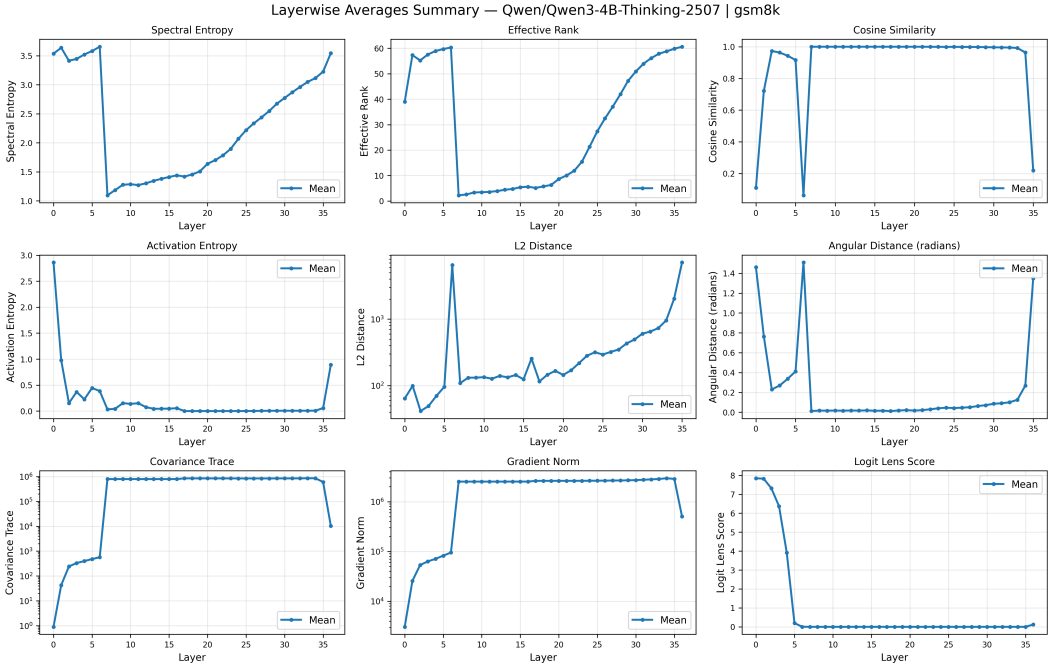


Figure 14: Layerwise Average Summary - Qwen3-4B-Thinking-2507 on GSM8K

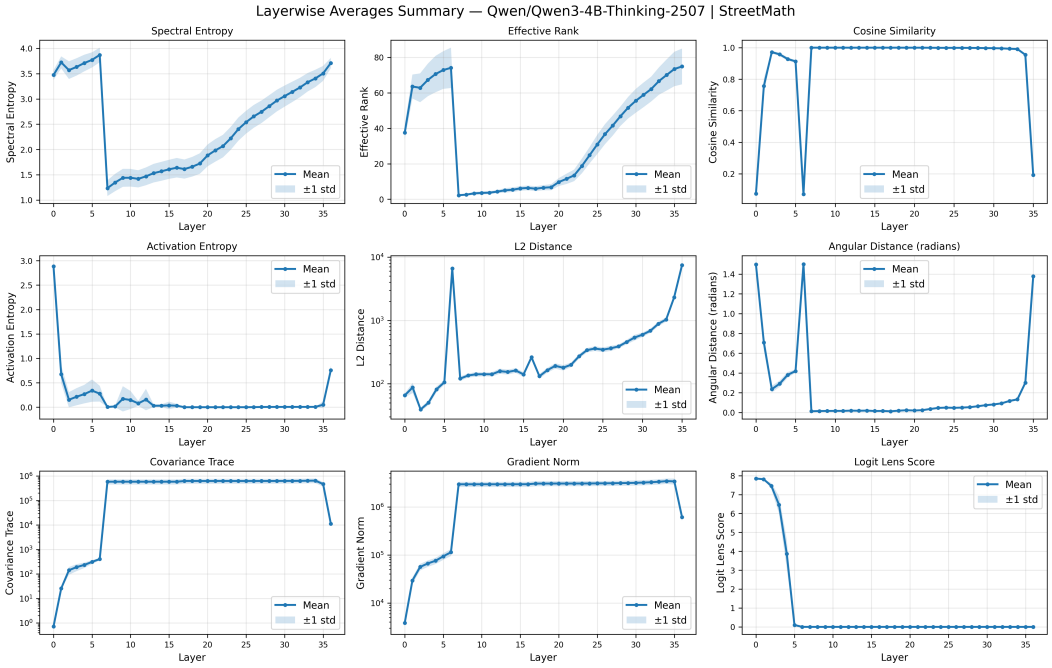


Figure 15: Layerwise Average Summary - Qwen3-4B-Thinking-2507 on StreetMath

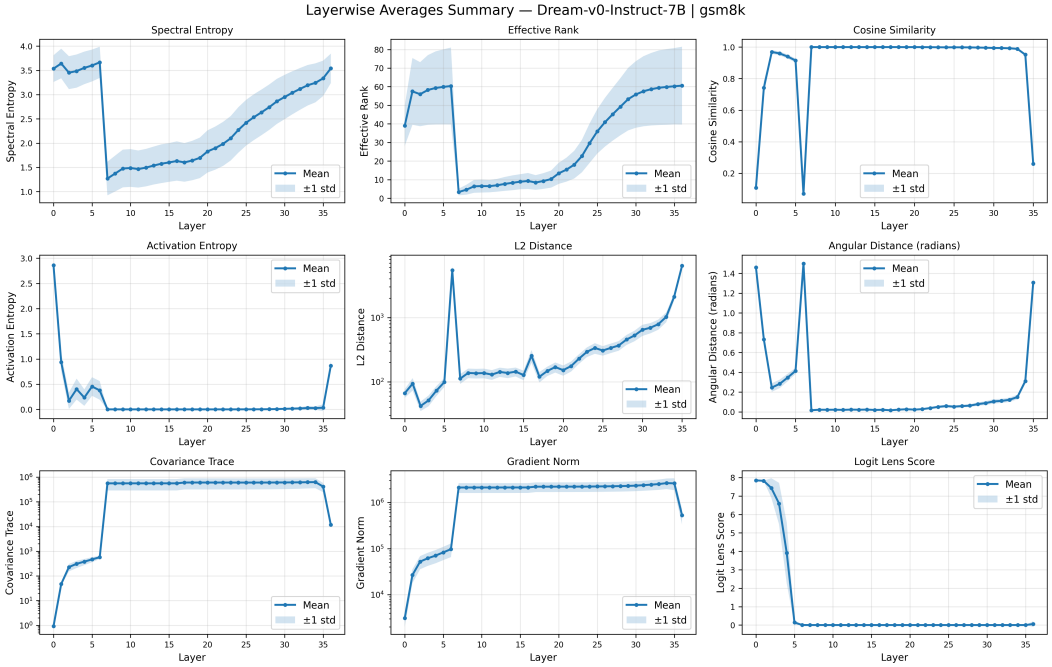


Figure 16: Layerwise Average Summary - Dream-v0-Instruct-7B on GSM8K

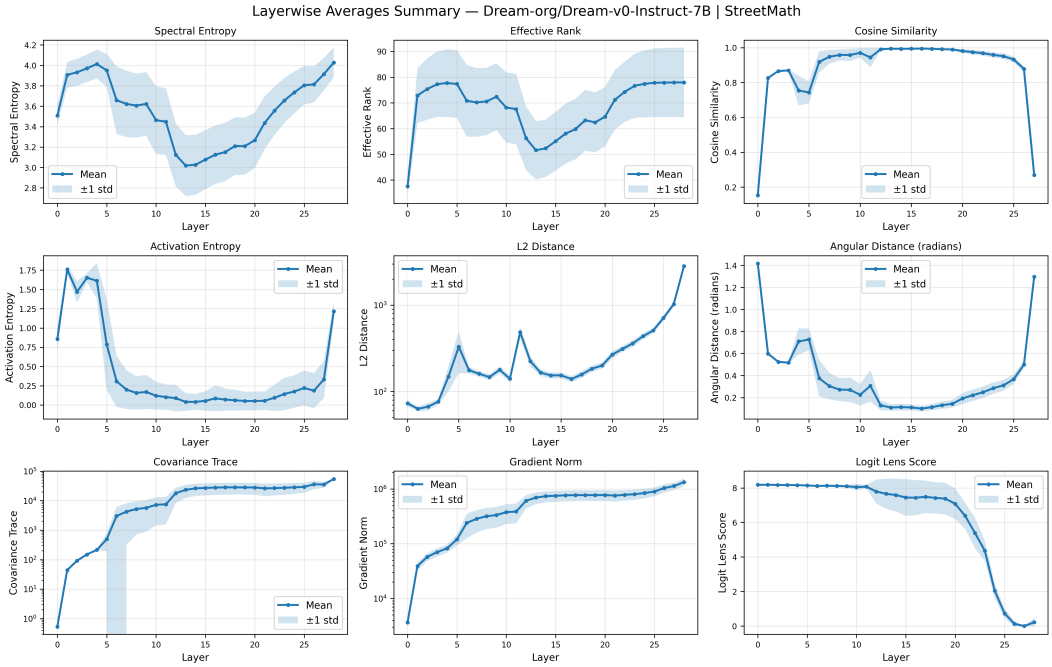


Figure 17: Layerwise Average Summary - Dream-v0-Instruct-7B on StreetMath

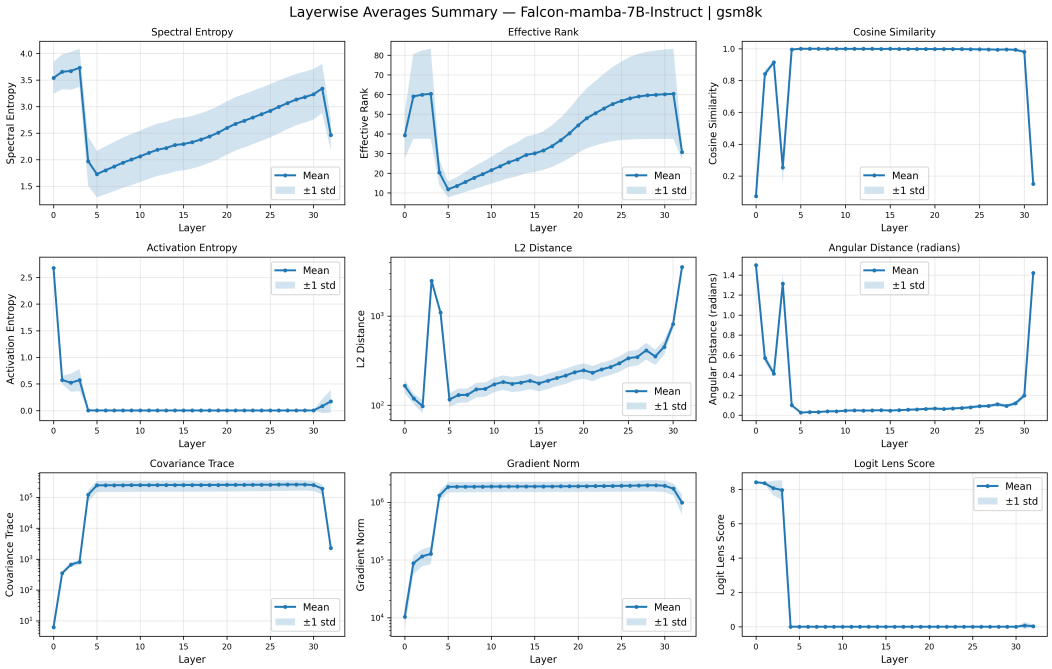


Figure 18: Layerwise Average Summary - Falcon-mamba-7B on GSM8K

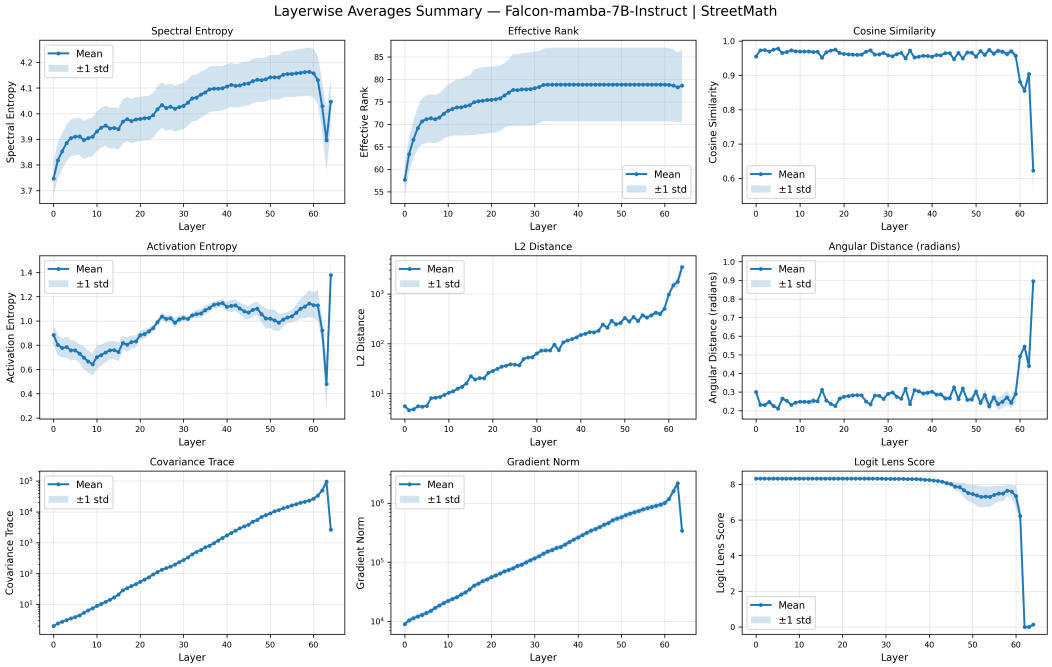


Figure 19: Layerwise Average Summary - Falcon-mamba-7B on StreetMath

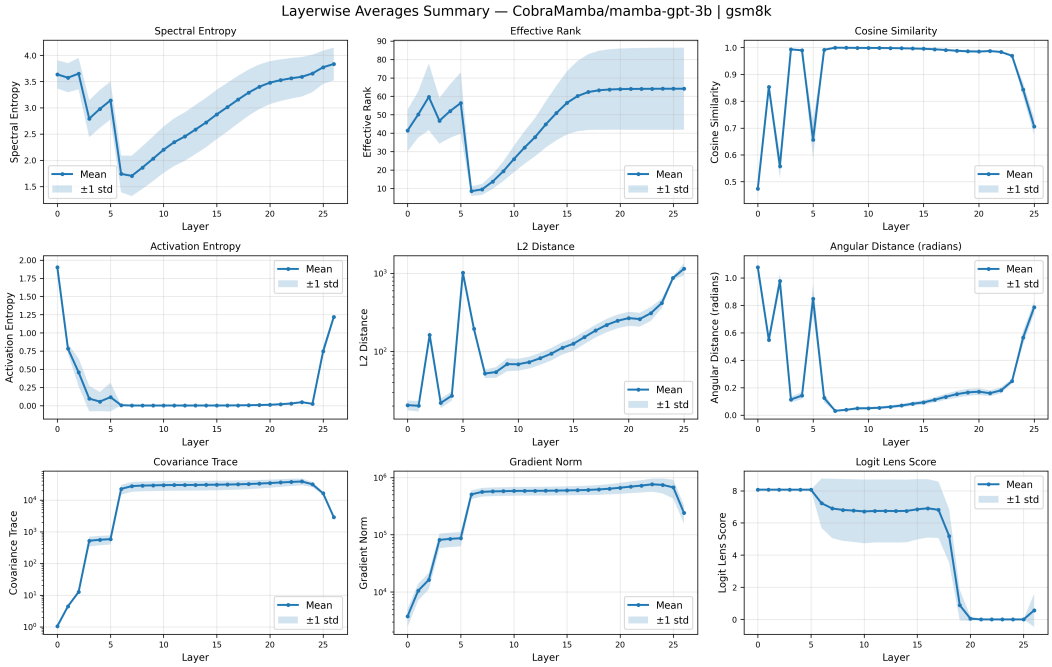


Figure 20: Layerwise Average Summary - mamba-gpt-3B on GSM8K

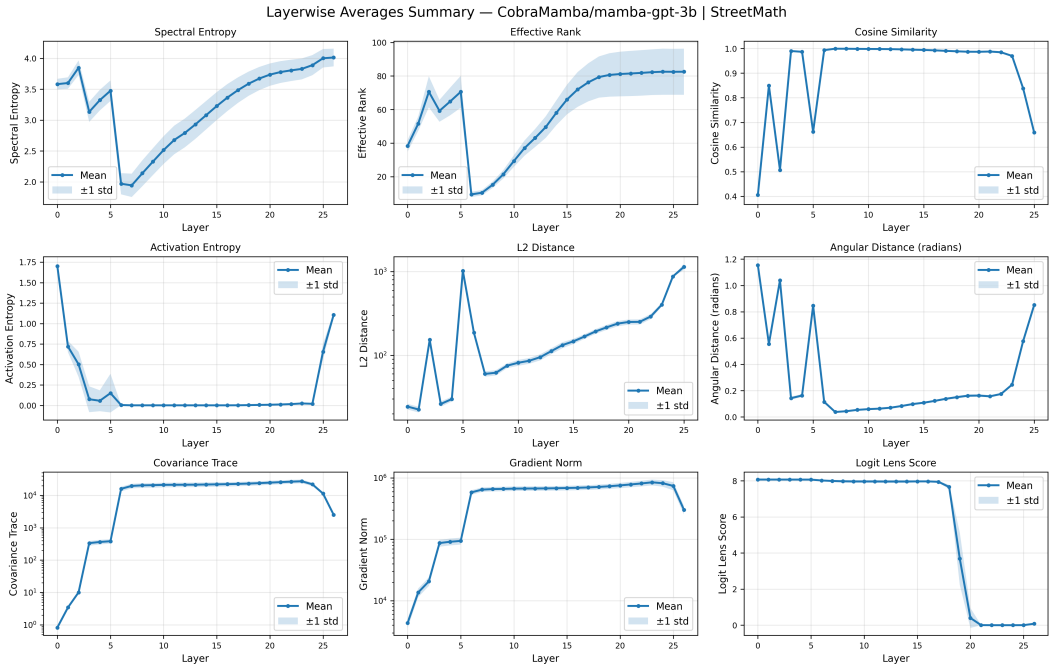


Figure 21: Layerwise Average Summary - mamba-gpt-3B on StreetMath

F.2 Training Data Bias Toward Exact Computation

Research reveals systematic biases in mathematical reasoning training data that favor exact computation over flexible approximation strategies. Analysis of major mathematical training corpora shows a predominant focus on problems with exact, verifiable answers. Paster et al.'s OpenWebMath dataset (28), containing 14.7B tokens of mathematical web content, consists primarily of forum discussions, educational materials, and reference pages where mathematical problems are presented with definitive solutions rather than approximation strategies. Similarly, Lewkowycz et al.'s Minerva training corpus (29) drew from 118GB of scientific papers and mathematical web content that emphasizes precise computational procedures.

This training bias toward exact answers has measurable consequences for model behavior. The pattern-matching hypothesis is supported by Mirzadeh et al.'s GSM-Symbolic analysis (30), which reveals that model performance degrades significantly when numeric values are perturbed, indicating over-reliance on specific number patterns rather than general reasoning principles. Shao et al. (31) explicitly acknowledge this issue, noting that their model exhibits "data selection bias in pre-training and fine-tuning" that leads to weaker performance on certain problem types.

F.3 Overthinking and Computational Inefficiency

Recent work has documented a troubling pattern: LLMs consistently overthink mathematical problems, generating verbose reasoning chains when simpler approaches would suffice. Ding et al. (32) proposed "break the chain" strategies to reduce token consumption, demonstrating that models maintain performance even when forced to skip intermediate steps. Zhao et al.'s work on efficiency enhancement in reasoning models (33) suggests this isn't just a performance issue but a fundamental architectural limitation.

F.4 Mechanistic Evidence for Competing Circuits

Mechanistic interpretability studies reveal distinct and overlapping neural pathways for exact versus approximate reasoning. Christ et al. (9) demonstrated that math-specific parameters can be isolated through structured pruning. Skean et al. (34) conducted a layer-by-layer analysis, revealing that different types of mathematical operations are processed at different depths in transformer architectures.

Sun et al. (35) probed arithmetic errors in language models and identified systematic patterns in computational failures, while Saynova et al. (36) investigated whether mathematical reasoning relies on fact recall, heuristics, or pure computation, finding evidence for multiple pathways depending on problem complexity and context.

F.5 Numerical Representation and Geometric Understanding

Understanding how LLMs represent numerical information has been a focus of recent mechanistic interpretability work. Levy and Geva (8) demonstrated that language models encode numbers using individual circular representations for each digit in base 10, providing geometric understanding of numerical processing. Kantamneni and Tegmark (37) extended this work by showing that language models use trigonometric functions in their internal computations, suggesting sophisticated geometric representations of numerical concepts. Zhu et al. (38) investigated how language models encode numeric magnitude, while Shah et al. (39) examined magnitude comparison tasks, finding that models develop specialized circuits for determining relative numerical size. These representational studies suggest that current numerical encodings may be too rigid to support flexible approximation strategies.

F.6 Architectural Differences in Approximation Capacity

Different LLM architectures exhibit varying capabilities for flexible reasoning, though systematic evaluation of approximation strategies across architectures remains limited. Li et al. (40) explored diffusion models for language tasks, demonstrating their application to text generation, though their mathematical reasoning capabilities, particularly regarding approximation versus precision trade-offs, have not been extensively studied.

The architectural constraints that affect mathematical reasoning extend beyond approximation to fundamental information processing capabilities. Jelassi et al. (41) demonstrated that transformers can theoretically copy strings of exponential length while state-space models are fundamentally limited by their fixed-size latent state, suggesting that the rigid memory constraints that impede copying may also constrain flexible approximation strategies. These findings indicate that current architectural paradigms may systematically differ in their capacity for the kind of cognitive flexibility that characterizes human mathematical reasoning.

This architectural variation highlights a broader gap in our understanding of how different model designs affect the ability to engage in contextually appropriate approximation—a crucial aspect of mathematical intelligence that remains largely unexplored across the spectrum of current LLM architectures.

F.7 Augmentation Strategies and Alternative Approaches

Recognizing the limitations of pure language model approaches to arithmetic, researchers have proposed several augmentation strategies. Tool-augmented approaches represent the dominant paradigm, where models learn to invoke external calculators, symbolic solvers, or knowledge bases. Schick et al. (42) introduced Toolformer, which teaches LLMs to use tools through self-supervised learning, while Das et al. (43) developed MathSensei, combining web search, Python execution, and Wolfram-Alpha integration for comprehensive mathematical reasoning support.

Program-aided reasoning offers another promising direction. Gao et al. (44) proposed Program-Aided Language models (PAL), which generate Python programs as intermediate reasoning steps, while Chen et al. (45) introduced Program-of-Thoughts prompting to separate computation from reasoning. These approaches effectively delegate precise calculations to programming environments while preserving natural language reasoning.

At the architectural level, Dietz and Klakow (46) introduced the Integrated Gated Calculator (IGC), which emulates a calculator directly on the GPU, achieving 98-99% accuracy on arithmetic tasks in a single iteration without external tools. Lauter et al. (47) investigated machine learning approaches for modular arithmetic, demonstrating specialized techniques for specific algebraic structures, though with limited success that highlights the inherent difficulty of certain mathematical operations.

While these augmentation strategies successfully address computational limitations and improve exact calculation capabilities, they do not resolve the fundamental issue our work identifies: the inability to

engage in contextually appropriate approximation when exact computation is unnecessary. Current approaches actually reinforce the precision bias by providing increasingly sophisticated mechanisms for exact calculation, potentially exacerbating the cognitive inflexibility that characterizes current mathematical reasoning systems.

F.8 Pattern Recognition vs. Algorithmic Understanding

A fundamental question concerns whether models learn genuine algorithms or rely on sophisticated pattern recognition. Nikankin et al. (48) examined "arithmetic without algorithms," investigating whether models can perform mathematical reasoning without explicit algorithmic procedures, suggesting that models may rely on pattern recognition and approximation strategies that differ fundamentally from formal mathematical computation. Gambardella et al. (49) investigated whether language models perform hard arithmetic by examining their computational processes, while Lovering et al. (50) examined language model probabilities in mathematical contexts, providing insights into how models represent uncertainty and confidence.

F.9 The Need for Approximation-Aware Evaluation

Current mathematical reasoning evaluation focuses exclusively on exact computation, creating a fundamental evaluation gap that obscures crucial aspects of mathematical intelligence. While Ahn et al.'s comprehensive survey (51) emphasizes that "accuracy shouldn't be the sole metric" for evaluating mathematical reasoning and highlights the need for more robust evaluation beyond final-answer correctness, existing benchmarks continue to reward only precise answers regardless of contextual appropriateness.

This evaluation paradigm fails to assess whether LLMs can engage in the kind of flexible, context-appropriate approximation that characterizes human mathematical cognition in everyday settings. The gap is significant because it touches on fundamental questions about the nature of machine intelligence and whether current LLMs genuinely understand mathematical concepts or merely implement sophisticated pattern matching. Without evaluating approximation capabilities, we cannot determine if models possess the cognitive flexibility necessary for human-like mathematical reasoning in diverse contexts.

G Limitations

While our work provides new insights into the approximation behavior of LLMs, several limitations remain. First, the *StreetMath* dataset contains only 1,000 problems, which may not capture the full variety of real-world estimation tasks. Second, our evaluation focuses on a specific set of open-source models; results may not generalize to larger proprietary systems or other architectures. Third, our analysis is restricted to numerical approximation in simple arithmetic settings. Extensions to more complex mathematical domains are left for future work.

Acknowledgments

We acknowledge the use of AI tools (ChatGPT, Codex) for text proofreading, formatting assistance and scripting.