SENSEMATH: DO LLMS HAVE NUMBER SENSE? EVALUATING SHORTCUT USE, JUDGMENT, AND GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models often default to step-by-step computation even when efficient numerical shortcuts are available. This raises a basic question: do they exhibit *number sense* in a human-like behavioral sense, i.e., the ability to recognize numerical structure, apply shortcuts when appropriate, and avoid them when they are not? We introduce SENSEMATH, a controlled benchmark for evaluating structure-sensitive numerical reasoning in LLMs. SENSEMATH contains 4,800 items spanning eight shortcut categories and four digit scales, with matched strong-shortcut, weak-shortcut, and control variants. In generating the items we follow the Cognitive Load Theory by manipulating the intrinsic load via number of digits and short-cut availability, while controlling the extraneous load by fixing the item type and format. SENSEMATH supports three evaluation settings of increasing cognitive complexity following the Bloom's Taxonomy: *Shortcut Use* (whether models can apply shortcuts on shortcut-amenable problems); *Applicability Judgment* (whether they can recognize when a shortcut is appropriate or misleading); and *Problem Generation* (whether they can generate new problem items that correctly admit a given type of shortcut). Our evaluation across six model configurations shows a consistent pattern: when explicitly prompted, models readily adopt shortcut strategies and achieve substantial accuracy gains on shortcut-amenable items (up to 15%), yet under standard chain-of-thought prompting they spontaneously employ such strategies in fewer than 40% of cases, even when they demonstrably possess the requisite capability. Moreover, models systematically *over-generalise* shortcuts to problems where they do *not apply*, and fail to *generate* valid shortcut-bearing problems from scratch. Together, these results suggest that current LLMs exhibit *procedural* shortcut fluency without the *structural* understanding of when and why shortcuts work that underlies human number sense. Our code and data are available at `[anonymous repository]`.

1 INTRODUCTION

Recent advances in large language models (LLMs) have led to remarkable progress on arithmetic and mathematical reasoning tasks Chain-of-thought (CoT) prompting (Wei et al., 2022; Kojima et al., 2022) and reinforcement-learning-based approaches such as OpenAI o1 (OpenAI, 2024) and DeepSeek-R1 (Guo et al., 2025) have pushed performance on mathematical benchmarks (e.g., GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021)) to near-human levels. However, it remains unclear whether this success reflects a genuine understanding of numerical structure.

A central concept capturing human mathematical reasoning ability is *number sense*: the capacity to perceive structure in numbers, flexibly choose solution strategies (e.g., efficient computational strategies), and generalize them across contexts (McIntosh et al., 2005). For example, when computing 98×14, a human can exploit proximity to 100 and compute $(100 - 2) \times 14$, whereas for 73×14, no such shortcut naturally applies and a different method is used. Likewise, when comparing 10/11 and 11/12, a human may avoid cross-multiplication altogether by observing that both fractions are close to 1 and that 11/12 is larger because it has the smaller gap to 1.

In developing and evaluating number sense, it is important to assess not only whether learners can *apply* number sense strategies, but also whether they can engage with them at *higher cognitive levels*.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Following Bloom's Taxonomy (Bloom et al., 1956; Krathwohl, 2002), a widely used framework in education for assessing different levels of cognitive complexity, these higher levels include *application, evaluation*, and *creation*. Learners should be able to apply number sense strategies in problem solving, judge when a number sense strategy is appropriate, and generate new problems that meaningfully involve number sense. These abilities reflect deeper and more advanced understanding. This view is consistent with research that frames number sense as a flexible, adaptive form of mathematical reasoning rather than a fixed set of procedures (Boaler, 2022; Devlin, 2010), highlighting the importance of how learners select, justify, and generate strategies in context.

Despite their importance, current evaluations of LLM reasoning do not directly test these capabilities. Most benchmarks focus on answer correctness (Cobbe et al., 2021; Hendrycks et al., 2021; Lightman et al., 2024; Art of Problem Solving, 2024), while studies of CoT plausibility examine whether reasoning traces faithfully reflect the model's internal computation (Turpin et al., 2023; Lanham et al., 2023; Paul et al., 2024). While useful, these metrics primarily assess whether a model can execute a reasoning procedure, and do not distinguish between *procedural competence* and a *deeper, declarative understanding* of numerical structure.

Numerical reasoning problems can become challenging for reasons beyond the arithmetic operation itself. Two problems may require the same basic computation, yet differ in how many quantities, relations, or intermediate results must be tracked before reaching the answer. Here, cognitive load denotes the amount of intermediate state a solver must maintain and manipulate during reasoning, rather than merely the surface difficulty of the problem.

This limitation is particularly relevant in *educational* settings, where LLMs are increasingly used as AI tutors. Effective teaching requires more than demonstrating how to solve a problem: it involves helping students understand when a strategy is applicable, when it is not, how to recognize the underlying structure, and generating new problems for students to practice these distinctions.

Motivated by such gaps in current evaluations of mathematical reasoning, we study whether LLMs exhibit behaviors consistent with number sense through three progressively more demanding capabilities, formalized as the following research questions:

- RQ1: Do LLMs invoke such shortcuts spontaneously in the absence of explicit instruction?
- RQ2: In cases where shortcuts are not invoked spontaneously, can LLMs apply them when explicitly instructed?
- RQ3: Do LLMs misuse shortcuts when they appear applicable but are not?
- RQ4: Can LLMs generate new problems that admit a given type of shortcut?

These RQs correspond to increasing cognitive demands: from ***applying*** a known method (RQ1,RQ2), to ***judging*** when it should be used (RQ3), to ***creating*** new items (RQ4). Despite its importance, evaluating number sense in LLMs is inherently challenging. Unlike standard mathematical reasoning, number sense is not a single well-defined skill, but rather a flexible meta-cognitive disposition (Sowder, 2020). As a result, valid assessment requires carefully constructed instruments that go beyond accuracy on isolated tasks and instead probe behavior across controlled variations (Kirkland et al., 2024). Existing efforts provide only a partial view of this capability. While LLMs can solve complex reasoning problems, they often fail on basic numerical operations such as magnitude comparison and digit manipulation (Yang et al., 2024). Prior evaluations primarily target narrow, perception-level aspects of numerical ability: for instance, Testolin (2024) study elementary numerical discrimination, and Rahman & Mishra (2025) evaluate combinatorial reasoning over numbers using tasks such as the Game of 24. Although informative, these tasks largely assess whether a model can *perceive* numerical properties, rather than whether it can *reason with them*, in particular, whether it can selectively apply efficient strategies when the problem structure permits.

To bridge this gap, we introduce SENSEMATH, a controlled benchmark designed to evaluate structure-sensitive numerical reasoning. The benchmark consists of 4,800 items spanning eight categories and four digit scales ($d \in 2, 4, 8, 16$), covering both problem-level shortcuts (e.g., magnitude estimation, structural decomposition, relative distance, cancellation, compatible numbers, and landmark comparison), reasoning-level shortcuts (equation-based transformations), and option-level shortcuts (e.g., elimination strategies). Each item has matched strong-shortcut, weak-shortcut, and control variants, enabling causal attribution of performance differences to strategy selection rather than underlying problem difficulty. The items included in SENSEMATH were generated following the Cognitive Load

Theory by varying the intrinsic load while controlling the extraneous load. This design allows us to systematically test whether models exploit available structure, avoid overgeneralizing shortcuts, and remain stable when such structure is absent. An overview of the benchmark is shown in Figure 1.

Key findings. We evaluate six model configurations (GPT-4o-mini, GPT-4.1-mini, Qwen3-30B, Qwen3-8B, Llama-3.1-8B, and o3) on SENSEMATH. These evaluations span three levels of increasing cognitive demand: *Shortcut Use*, *Applicability Judgment*, and *Problem Generation*. Our findings map directly to the research questions above.

- **RQ1: Sometimes.** Under standard CoT prompting, shortcut strategies appear in fewer than 40% of responses at $d{=}4$, though this rate increases with digit scale.
- **RQ2: Yes, but more for capable models.** When explicitly instructed, capable models achieve accuracy gains of up to 15% (GPT-4.1-mini at $d{=}8$), while 8B-parameter models show no benefit or even degradation.
- **RQ3: Yes, severely misused.** Models accept nearly all problems as shortcut-amenable (control rejection rate as low as 0%) and over-apply estimation strategies, reducing accuracy on control items by up to 12%.
- **RQ4: Largely failed in generation.** Models construct valid shortcut-amenable problems at only 0–21.9% pass rate, reproducing the surface form but not the structural constraints that make a shortcut effective.

Contributions. This work makes the following contributions:

- We introduce a new perspective for evaluating numerical reasoning in LLMs through the lens of number sense, organizing evaluation by the level of cognitive engagement following Bloom's revised taxonomy (Krathwohl, 2002), spanning Apply (*Shortcut Use*), Analyze (*Applicability Judgment*), and Create (*Problem Generation*).
- We instantiate this framework with SENSEMATH, a controlled benchmark of 4,800 items designed to isolate structure-sensitive behavior. Its construction is guided by cognitive load theory (Sweller, 1988): items vary along two orthogonal dimensions of intrinsic cognitive load: (1) number of digits and (2) number-sense strategy applicability, while extraneous load is held constant through a shared surface template (Ober et al., 2023).
- Empirically, we show that some LLMs can execute shortcuts when prompted but most of them struggle to use them appropriately or generate valid new items.

Impact: Our results reveal a gap between procedural execution and structural numerical understanding in LLMs: models may be able to execute shortcut-based solutions when explicitly prompted, but are substantially less reliable at judging when such strategies apply or at generating new examples. Since these abilities are important for supporting flexible mathematical reasoning, our findings suggest caution when deploying LLMs as autonomous math tutors.

2 RELATED WORK

Number sense and its assessment in LLMs. The concept of number sense originates in mathematics education: McIntosh et al. (2005) define it as a person's general understanding of number and operations together with the inclination to use this understanding flexibly, while McIntosh et al. (2005) identify core components such as magnitude estimation, benchmark use, and recognition of numerical structure. Sowder (2020) further characterise number sense as a meta-cognitive disposition to choose efficient computational paths, and Yang (2003) operationalise these ideas into validated instruments for school-age children. Recent work has begun probing analogous abilities in LLMs. Yang et al. (2024) show that models excel at complex reasoning yet fail on basic numerical operations like magnitude comparison and digit manipulation. Rahman & Mishra (2025) find strong deterministic performance but failure on heuristic-search tasks such as the Game of 24, suggesting a "fragile number sense." Li et al. (2025) expose systematic gaps across six numeracy capabilities, and Nikankin et al. (2024) reveal that LLMs solve arithmetic via sparse neuron-level heuristics rather than learned algorithms. However, these efforts probe only narrow, perception-level facets of number sense. SENSEMATH goes further by testing whether models can *selectively activate* shortcut strategies through prompting, and evaluates number sense at three cognitive levels (*Apply - Shortcut Use, Evaluation - Applicability Judgment*, and *Create - Problem Generation*).

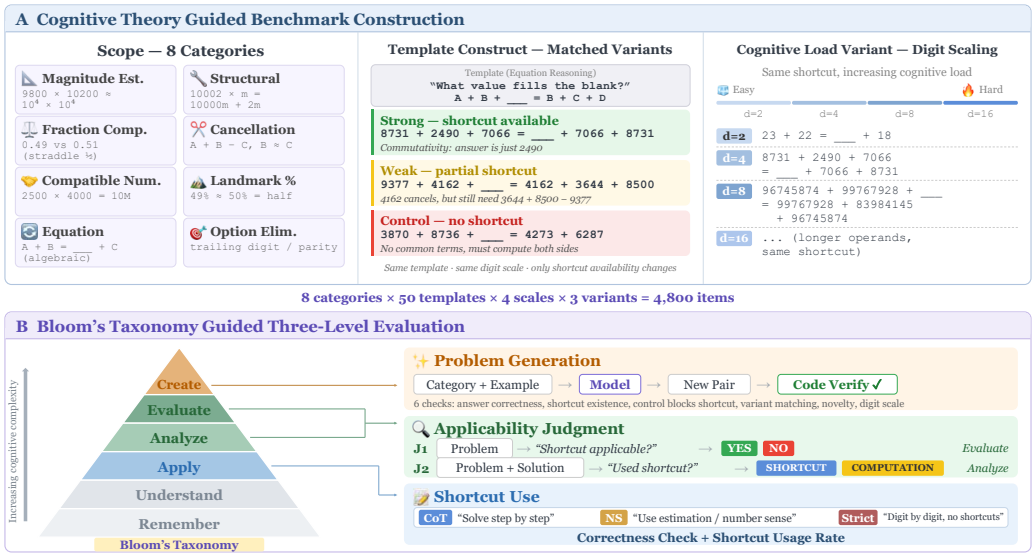

Figure 1: SENSEMATH overview. (A) Benchmark construction spans eight shortcut categories, matched strong-shortcut, weak-shortcut, and control variants, and four digit scales. (B) Evaluation is aligned with Bloom's revised taxonomy: Apply corresponds to Shortcut Use under CoT, number-sense, and Strict prompting; Evaluate and Analyze correspond to applicability judgment (J1) and strategy identification (J2); Create corresponds to Problem Generation.

LLM reasoning and evaluation. Chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022) and its variants (Wang et al., 2022; Yao et al., 2023) have substantially improved LLM performance on mathematical benchmarks such as GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), MathBench (Liu et al., 2024), and MathVista (Lu et al., 2023). Meanwhile, a parallel line of work questions the reliability of these reasoning traces: Turpin et al. (2023) demonstrate that CoT explanations can be systematically unfaithful, and diagnostic benchmarks such as GSM-Symbolic (Mirzadeh et al., 2024) and MATH-Perturb (Huang et al., 2025) expose fragilities behind high aggregate scores. On the efficiency side, methods like Chain of Draft (Xu et al., 2025), Sketch-of-Thought (Aytes et al., 2025), and Token-Budget-Aware reasoning (Han et al., 2025) aim to compress verbose reasoning, while Chen et al. (2024) show that O1-like models waste computation on trivial problems. These approaches share a common assumption: the underlying strategy is fixed, and the goal is to express it more concisely. By contrast, number sense involves switching to a qualitatively *different strategy* that exploits numerical structure to bypass unnecessary computation. Our work is therefore complementary: existing methods compress the *same* strategy; SENSEMATH measures if models can *recognize when a different shortcut-based strategy is available and use it*.

3 THE SENSEMATH BENCHMARK

This section describes our design philosophy, category taxonomy, programmatic generation pipeline, and evaluation framework.

3.1 DESIGN PHILOSOPHY

We design our evaluation around how humans exhibit *number sense* in arithmetic: not merely by producing correct answers, but by recognizing numerical structure, selecting efficient shortcuts when appropriate, avoiding them when they are not, and generating new examples for further practice. Our goal is therefore not to test whether LLMs can compute numerical answers by any means available, but whether they can reason in a way that reflects structure-sensitive shortcut use. We do not argue that LLMs should always solve math problems in this way. In many applications, models can directly obtain correct answers by calling external tools such as calculators or programs. That, however, is not the focus of this work. Instead, we study number sense from an educational perspective: whether LLMs can recognize when a shortcut applies, explain it clearly, avoid misusing it, and generate new examples for students to practice.

Shortcut is a reasoning strategy that exploits numerical structure to simplify or bypass standard step-by-step computation. Examples include using proximity to base values, cancellation, or relative comparisons, such as rewriting 98×14, as $(100 - 2) \times 14$, or comparing 10/11 and 11/12 by noting that both are close to 1 and comparing their gaps to 1. A transformation does not qualify as a shortcut if the simplified form still requires essentially the same level of computation, for example, rounding $5,374 \times 2,169$ to $5,400 \times 2,200$ still leaves a multi-digit multiplication.

We follow Cognitive Load Theory (Sweller, 1988) to construct items with shortcut-based solutions and controlled variants. We hold extraneous load constant through 50 uniform surface templates and a fixed multiple-choice format, while varying intrinsic load along two orthogonal dimensions: digit scale ($d \in \{2, 4, 8, 16\}$) and shortcut availability (strong/weak/control), described next.

Shortcut-Invariant Scaling. Given a question template that admits a particular shortcut, we can generate multiple items by varying the digit scale while preserving the same underlying structural cue. This allows us to test whether models can maintain shortcut-based reasoning as surface numerical complexity increases. Importantly, the digit scale is designed to be *shortcut-invariant*: although the numbers become longer, the structural relation that licenses the shortcut remains unchanged. For example, the reasoning shortcut used to compare 10/11 and 11/12 is identical to that used for comparing 1110/1111 and 1111/1112 (comparing their gaps to 1), despite the larger number of digits.

Matched Shortcut Variants. For each problem template at a fixed digit scale, we construct **three matched variants**: ***strong***, where a clean and effective shortcut is available; ***weak***, where a shortcut is partially helpful but still requires additional computation; and ***control***, where no effective shortcut applies. These variants share the same template and digit scale and differ only in their numerical instantiation, ensuring that surface form and overall problem difficulty remain comparable. This matched design allows us to isolate the effect of shortcut availability and attribute performance differences to strategy selection rather than confounding factors.

3.2 Categories of Shortcuts

SenseMath includes 8 categories of shortcuts, organized into three tiers based on where the shortcut operates. These categories cover core forms of number-sense reasoning competencies identified in the mathematics education literature (McIntosh et al., 2005).

Tier 1: Problem-level shortcuts

1. *Magnitude estimation (ME)* tests the ability to approximate products by rounding operands to nearby powers of 10, which is foundational skill in gauging whether an answer is "in the right ballpark."
2. *Structural shortcuts (SS)* require recognising and exploiting algebraic identities near round numbers (e.g., $99 \times 37 = (100-1) \times 37$), probing whether models can decompose computations into simpler parts.
3. *Relative distance (RD)* measures relational reasoning about fractions by comparing their distance from a common benchmark (e.g., $1/2$), avoiding full cross-multiplication.
4. *Cancellation (CI)* tests sensitivity to near-cancellation patterns in expressions like $A + B - C$ when $B \approx C$, reducing a three-operand computation to simple subtraction.
5. *Compatible numbers (CN)* assesses whether models can identify product-friendly rounding opportunities (e.g., recognising that $248 \times 4{,}012 \approx 250 \times 4{,}000$), a key mental-math strategy.
6. *Landmark comparison (LC)* probes the use of familiar reference points when comparing percentages or fractions (e.g., $49\% \approx 50\%$), reflecting the human tendency to anchor judgments to well-known values.

Tier 2: Reasoning-level shortcut. *Equation reasoning (ER)* moves beyond arithmetic to algebraic structure: recognising identities such as commutativity and common-term cancellation in fill-in-the-blank equations, reducing multi-step algebra to a single structural observation.

Tier 3: Option-level shortcut. *Option elimination (OE)* tests a meta-reasoning ability: ruling out answer choices by quick feasibility checks (trailing digit, parity, order-of-magnitude) without computing the exact answer, reflecting strategic use of multiple-choice format itself.

For each category, we generate 50 question templates. Each templates is instantiated at four digit scales ($d \in \{2, 4, 8, 16\}$) through *Shortcut-Invariant Scaling*, and each scaled item is further expanded

into three *Matched Shortcut Variants* (*strong*, *weak*, and *control*). In total, this yields $8 \times 50 \times 4 \times 3 = 4{,}800$ problem items. Example items for each category are provided in Appendix A. All items are generated programmatically via category-specific Python generators with rejection sampling to guarantee answer correctness and controlled distractor difficulty. Full generation details are provided in Appendix C.

3.3 Three-Level Evaluation

Usage of Number Sense Strategy. To answer RQ1 and RQ2, we evaluate models under three prompting conditions that vary in the extent to which they encourage shortcut-based reasoning. **CoT** (chain-of-thought) uses a standard step-by-step reasoning prompt, testing whether models invoke shortcuts spontaneously. **NS** (number-sense) encourages mathematical intuition and easy calculations, but does not mention any specific shortcut type, making it deliberately category-agnostic. **Strict** explicitly forbids shortcuts and requires fully explicit computation, serving as a negative control. All conditions require answers inside `\boxed{}` for unambiguous extraction. Full prompt templates are provided in Appendix B. We use two metrics to report model performance: **Accuracy**, which measures final-answer correctness, and **Shortcut Usage Rate**, which measures the proportion of responses that use shortcut-based reasoning on instances where such strategies are applicable.

Applicability Judgment. To evaluate whether models can distinguish between cases where shortcuts are appropriate and where they are not, we introduce two judgment tasks. **J1** presents a problem and asks whether it can be solved faster with shortcut reasoning (`YES`/`NO`). We evaluate J1 on 261 selected items. **J2** presents a problem together with a solution and asks whether the solution uses a shortcut or a standard computation strategy (`SHORTCUT`/`COMPUTATION`). J1 measures *recognition* of shortcut applicability, while J2 measures *identification* of the strategy type used.

Problem Generation. To test whether models can construct new items that admit a given shortcut, we ask each model, given a category description and one example, to generate a new *strong*/*control* problem pair. Generated items are verified using six code checks: mathematical and option validity, target-shortcut existence, target-shortcut blocking in the control, other-category shortcut blocking in the control, structural and digit matching, and novelty. This evaluates whether models can *construct* valid shortcut-amenable problems with the intended structural properties.

Full task specifications and verification details are provided in Appendix C.

4 Experiments

We organise our experiments around the research questions posed in the introduction: **RQ1:** Can LLMs invoke shortcuts spontaneously? (§4.2) **RQ2:** Can LLMs apply shortcuts when explicitly instructed? (§4.3) **RQ3:** Do LLMs misuse shortcuts when they are not appropriate? (§4.4) **RQ4:** Can LLMs generate new shortcut-amenable problems? (§4.5)

4.1 Experimental Setup

Models. We evaluate six model configurations: Qwen3-30B-A3B-Instruct-2507 and Qwen3-8B (Yang et al., 2025), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), GPT-4o-mini and GPT-4.1-mini (Hurst et al., 2024), and o3. GPT-4.1-mini is a more capable successor to GPT-4o-mini that achieves near-ceiling CoT accuracy on SenseMath at $d{=}4$, providing an important reference point for how strong baseline performance constrains the prompting asymmetry. Open-weight models are served via vLLM with tensor parallelism; GPT models are accessed through the OpenAI API. We set temperature to 0 except for o3, for which temperature is not configurable through the interface used. The Qwen, Llama, GPT-4o-mini, and GPT-4.1-mini configurations use `max_tokens`$=512$. To accommodate reasoning, we request an output-token budget of 16,384 tokens for o3 in non-Strict settings. We use `gpt-5.6-luna` with low reasoning effort to judge shortcut usage (SU), referred to as the Luna-low judge.

4.2 RQ1: Can LLMs Invoke Shortcuts Spontaneously?

CoT rarely triggers shortcuts. Figure 2 shows the shortcut usage (SU) rate on strong-shortcut items at $d{=}4$ under all three prompting conditions. Strict substantially suppresses shortcuts (SU 1.5–12.5%), supporting its role as a negative control. Under CoT, shortcuts appear in 26.3–58.8% of responses despite being available, indicating that models do not consistently invoke efficient alternatives spontaneously. NS prompting raises SU to 41.8–95.8%, an increase of 15.5–38.3pp across the five configurations. This gap suggests that number-sense reasoning is *available* in these

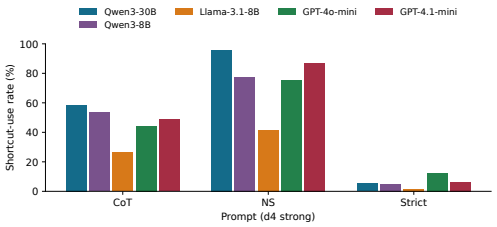

Figure 2: Shortcut-use rate on assessed strong items at d4 across prompting conditions, using the Luna-low judge. Conditions without assessed responses are not plotted.

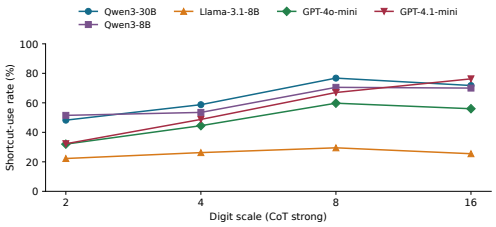

Figure 3: Shortcut-use rate on assessed strong items under CoT, by digit scale. Scales without assessed responses are not plotted.

Configuration	J1 acc.	J2 acc.
Qwen3-30B	55.9	85.0
Qwen3-8B	54.4	73.8
Llama-3.1-8B	52.5	86.2
GPT-4o-mini	54.8	87.5
GPT-4.1-mini	53.3	82.5
o3	55.2	80.0

Table 1: Judge accuracy (%) on the full paper evaluation sets: 261 J1 items and 80 J2 items. Reference labels combine benchmark labels with available human corrections.

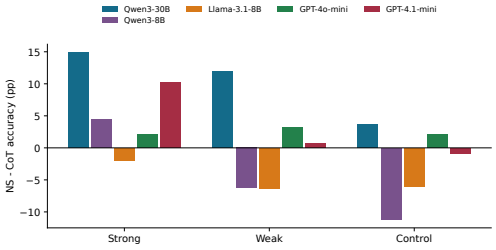

Figure 4: NS minus CoT accuracy (percentage points), pooled across digit scales by variant. Each prompt is scored on its evaluated responses.

models but is more consistently invoked with explicit prompting. Furthermore, the SU rate under CoT generally increases with digit scale (Figure 3): at d=16, CoT elicits shortcuts in 25.5–76.3% of responses, although the trend is not monotonic for every model.

NS benefit can increase with problem difficulty. Figure 5(a) shows that the accuracy gain from NS prompting can increase with digit scale, but the trend varies across models. GPT-4.1-mini gains +0.3pp at d=2 and peaks at +18.8pp at d=8; Qwen3-30B improves from +4.0pp at d=2 to +23.0pp at d=16. This is consistent with larger digit scales making brute-force computation more demanding, while structural shortcuts can reduce the required computation. Smaller models also benefit at some scales: Qwen3-8B gains +8.8pp at d=8, and Llama-3.1-8B gains +4.3pp at d=16, indicating that model size alone does not determine whether NS prompting helps.

4.3 RQ2: Can LLMs Apply Shortcuts When Instructed?

More capable models use NS prompting effectively. Figure 4 reveals a clear asymmetry in how NS prompting affects different models and matched variants (strong/weak/control). GPT-4.1-mini benefits primarily on strong items (+10.3pp), with little change on weak (+0.8pp) and control items (−0.9pp). GPT-4o-mini and Qwen3-30B benefit across all three variants; Qwen3-30B gains more on strong items (+14.9pp) than on controls (+3.7pp). In contrast, Qwen3-8B improves on strong items (+4.4pp) but drops on weak (−6.3pp) and control items (−11.2pp), showing that benefiting from shortcut prompting does not ensure robustness when the intended shortcut is weakened or absent. Llama-3.1-8B degrades across all variants when pooled across digit scales, further illustrating that the benefits of NS prompting are model-dependent. Full per-scale results are in Table 5 (Appendix E).

Per-shortcut category analysis. The aggregate pattern masks substantial variation across shortcut types. Figure 5(b) shows the NS minus CoT accuracy difference, in percentage points, at d=8. Large gains occur in *relative distance* (RD): GPT-4.1-mini improves from 32% to 84% (+52pp), and GPT-4o-mini from 46% to 66% (+20pp), a category where benchmark comparison can replace costly cross-multiplication. *Option elimination* (OE) is a revealing case: GPT-4.1-mini gains +26pp, whereas Qwen3-8B loses 34pp despite gaining +54pp on RD. This contrast shows that success on one shortcut category does not imply reliable performance on another. *Magnitude estimation* (ME) shows a smaller gain for GPT-4o-mini (+4pp at d=8). Smaller models show both gains and losses

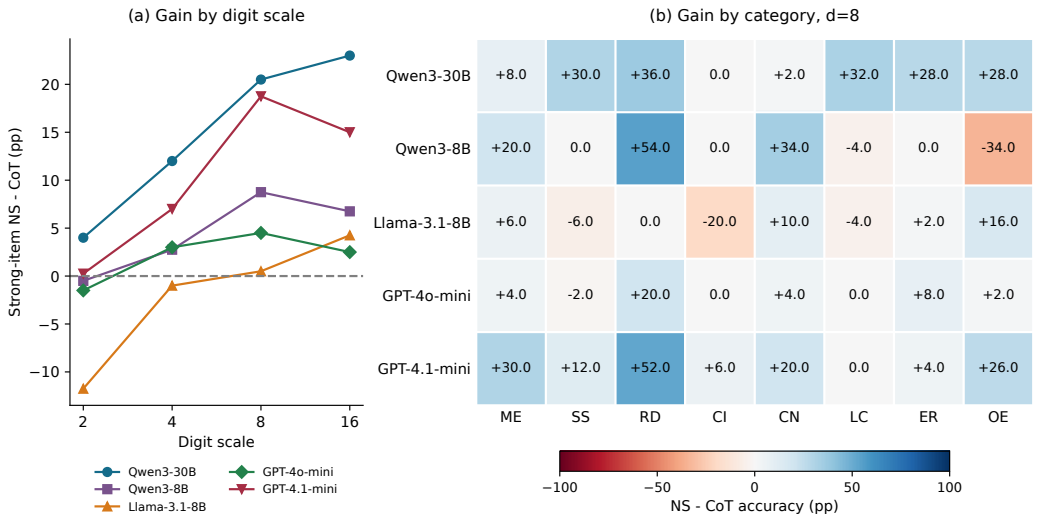

Figure 5: Prompt comparison on strong items: (a) NS minus CoT accuracy by digit scale; (b) NS minus CoT accuracy by category at d8. Both panels use percentage points. Each prompt is scored on its evaluated responses; N/A denotes an unavailable comparison.

Configuration	Math	Str.	Ctrl.	Cross	Match	Nov.	All
Qwen3-30B	19.8	22.9	14.6	2.1	45.8	40.6	0.0
Qwen3-8B	26.0	26.0	22.9	11.5	78.1	90.6	0.0
Llama-3.1-8B	12.5	6.2	12.5	3.1	71.9	57.3	0.0
GPT-4o-mini	18.8	28.1	16.7	5.2	71.9	100.0	0.0
GPT-4.1-mini	31.2	35.4	27.1	9.4	87.5	99.0	1.0
o3	45.8	40.6	43.8	30.2	93.8	100.0	21.9

Table 2: Automatic generation-check pass rates (%). Math: math/options; Str.: strong target; Ctrl.: control target blocked; Cross: other-domain shortcuts blocked; Match: matching/digits; Nov.: novelty.

across categories, rather than uniform degradation under NS prompting. Category heatmaps for other digit scales are provided in Appendix D.

4.4 RQ3: Do LLMs Misuse Shortcuts?

We score all 261 J1 items and 80 J2 items in the paper evaluation sets, incorporating available human corrections into the benchmark reference labels. For 49 J1 replacements without human review, we apply the original labeling rule (strong/weak: YES; control: NO).

Models often overapply shortcuts. As shown in Table 1, models show a strong YES bias in J1: GPT-4o-mini answers YES on all 261 items, while o3 rejects only 1 of the 91 control items. J1 accuracy ranges from 52.5% to 55.9% across the six configurations. In J2, where models classify whether a *given solution* used a shortcut, accuracy is higher (73.8–87.5%). Strategy identification is therefore more reliable than applicability judgment in this evaluation. (Judgment prompts are presented in Appendix B, and human validation in Section 4.6.)

4.5 RQ4: Can LLMs Generate Shortcut-Amenable Problems?

Generation largely fails. For generation, a check passes only when verified; all other outcomes are scored as failures. Table 2 shows that overall pass-all-6-checks rates range from 0% to 21.9%, with both the table and Figure 6 including all 96 generated pairs per model. Across all generated pairs, 8.3–35.4% pass four or five checks. o3 produces 21 pairs passing all six checks, compared with one for GPT-4.1-mini and none for the remaining models. A major obstacle is *math/options validity* (Math), with pass rates of 12.5–45.8%. These results show that partial success on individual checks does not ensure a valid strong/control pair; generation must satisfy mathematical validity, shortcut contrast, and the remaining construction requirements together.

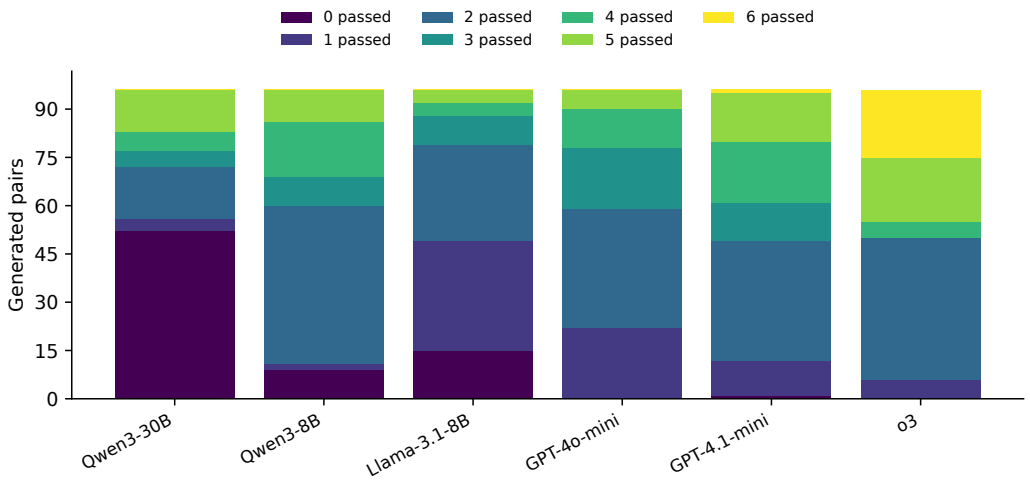

Figure 6: Distribution of automatic-check passes among generated pairs.

Implications from Evaluation. The Apply $\gg$ Analyze $>$ Create ordering suggests that current LLMs' number sense is largely *procedural*: they can execute shortcuts when instructed, but lack the declarative understanding needed to judge applicability or construct new instances. This dissociation parallels findings in human cognition where procedural fluency develops ahead of conceptual understanding (Rittle-Johnson et al., 2001).

4.6 Human Validation

We use human annotations to audit shortcut applicability labels and assess our automatic evaluators (Table 3). All annotators are adults with at least a bachelor's degree. Annotators assess whether a shortcut is available, whether a solution uses one, and whether a generated strong/control pair establishes the intended contrast. A shortcut must simplify computation from a human perspective; merely rewriting an expression or rounding operands without reducing the remaining computational burden does not qualify. Disagreements are adjudicated by a postdoctoral researcher in education to establish final reference labels. For J1, we audit 150 items from the full task pool and compare the strong/control assignments against human YES/NO judgments on 100 items (47 strong and 53 control). The remaining 50 weak items are excluded from this agreement calculation because partial shortcut availability does not imply a fixed binary label. SU agreement is measured on J2 responses after adjudication informed by the judge's predictions and explanations, rather than through independent blind validation. For G, the diagnostic sample contains 15 validator-pass and 15 validator-fail pairs.

Table 3: Agreement with final human references. For J1, strong/control assignments are mapped to YES/NO; n counts items used in each comparison.

Assessment	n	Agreement
J1 strong/control	100	97.0%
SU judge (J2)	65	89.2%
G validator	30	90.0%

5 Conclusion

We introduced number sense as a new lens for evaluating numerical reasoning in LLMs. We presented SenseMath, a controlled benchmark of 4,800 items designed to isolate structure-sensitive behavior by varying digit scale and shortcut availability under a common surface format. Our results show a clear gap between procedural and structural competence: LLMs can often execute shortcuts when explicitly prompted, but struggle to judge shortcut applicability and to generate new shortcut-amenable problems. This suggests that apparent success on arithmetic tasks may overestimate models' deeper numerical understanding.

AI USE STATEMENT

We used LLM-based assistants to support experiment planning and execution, write and debug code for data checking and repair, evaluation, and result analysis, and polish the manuscript. We also used an LLM-as-a-judge for shortcut-usage assessment, as described in the experimental setup. Validation included programmatic checks and targeted human annotation. The authors take responsibility for the final manuscript, results, and artifacts.

REFERENCES

Art of Problem Solving. AIME 2024 problems and solutions, 2024. https://artofproblemsolving.com/wiki/index.php/2024_AIME.

Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 24307–24331, 2025.

Benjamin S Bloom et al. Taxonomy of. *Educational Objectives*, 250(2025):3, 1956.

Jo Boaler. *Mathematical mindsets: Unleashing students' potential through creative mathematics, inspiring messages and innovative teaching*. John Wiley & Sons, 2022.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

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

Keith Devlin. The mathematical brain. *Mind, brain, and education*, 2010.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, et al. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. *arXiv preprint arXiv:2502.06453*, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Patrick K Kirkland, Ying Cheng, and Nicole M McNeil. A validity argument for a brief assessment of mature number sense. *Journal for Research in Mathematics Education*, 55(1):51–67, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

David R Krathwohl. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4): 212–218, 2002.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Haoyang Li, Xuejia Chen, Zhanchao Xu, Darian Li, Nicole Hu, Fei Teng, Yiming Li, Luyu Qiu, Chen Jason Zhang, Li Qing, et al. Exposing numeracy gaps: A benchmark to evaluate fundamental numerical abilities in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20004–20026, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2024.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6884–6915, 2024.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.

Alistair McIntosh, Barbara J Reys, and Robert E Reys. A proposed framework for examining basic number sense. In *Subject learning in the primary curriculum*, pp. 209–221. Routledge, 2005.

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

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. *arXiv preprint arXiv:2410.21272*, 2024.

Teresa M Ober, Yikai Lu, Chessley B Blacklock, Cheng Liu, and Ying Cheng. Development and validation of a cognitive load measure for general educational settings. *Journal of Psychoeducational Assessment*, 41(6):689–708, 2023.

OpenAI. Learning to reason with LLMs. *OpenAI Blog*, 2024.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15012–15032, 2024.

Roussel Rahman and Aashwin Ananda Mishra. A fragile number sense: Probing the elemental limits of numerical reasoning in llms. *arXiv preprint arXiv:2509.06332*, 2025.

Bethany Rittle-Johnson, Robert S Siegler, and Martha Wagner Alibali. Developing conceptual understanding and procedural skill in mathematics: An iterative process. *Journal of educational psychology*, 93(2):346, 2001.

Judith T Sowder. Making sense of numbers in school mathematics. In *Analysis of arithmetic for mathematics teaching*, pp. 1–51. Routledge, 2020.

John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2): 257–285, 1988.

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.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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, 2022.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Der-Ching Yang. Teaching and learning number sense–an intervention study of fifth grade students in taiwan. *International Journal of Science and Mathematics Education*, 1(1):115–134, 2003.

Haotong Yang, Yi Hu, Shijia Kang, Zhouchen Lin, and Muhan Zhang. Number cookbook: Number understanding of language models and how to improve it. *arXiv preprint arXiv:2411.03766*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Category	Strong	Control
SS	What is 98 * 34? (A) 3322 (B) 3342 **(C) 3332** (D) 3352	What is 84 * 63? **(A) 5292** (B) 5022 (C) 5112 (D) 5202
CI	Evaluate: 71 + 28 - 27 (A) 82 (B) 92 **(C) 72** (D) 62	Evaluate: 62 + 23 - 82 (A) 93 **(B) 3** (C) 273 (D) 183
ER	What value fills the blank? 23 + 22 = ___ + 24 (A) 31 **(B) 21** (C) 41 (D) 11	What value fills the blank? 20 + 93 + ___ = 59 + 49 (A) 15 (B) -15 **(C) -5** (D) 5
OE	What is 70 × 16? (A) 1123 (B) 1119 (C) 1121 **(D) 1120**	What is 54 * 17? (A) 828 (B) 1008 (C) 1098 **(D) 918**

Table 4: Matched d2 strong and control examples. Correct choices are bold. Control items pass the registered shortcut checks.

A Example Items

Table 4 shows example SenseMath items at d=2 for four categories.

B Prompt Templates

Use-level prompts. Three prompting conditions for solving SenseMath items:

CoT: Chain-of-thought

Please solve the following multiple-choice problem. Show your step-by-step reasoning, then provide your final answer as a single capital letter (A, B, C, or D) inside \boxed{}.

NS: Number-sense prompting

Please solve the following multiple-choice problem using only easy calculations. Rely on your mathematical intuition, number sense, and estimation. Show brief reasoning, then provide your final answer as a single capital letter (A, B, C, or D) inside \boxed{}.

Strict: Step-by-step computation

Please solve the following multiple-choice problem by performing precise, step-by-step arithmetic. Do NOT use estimation, rounding, or shortcuts. Show every computation digit by digit, then provide your final answer as a single capital letter (A, B, C, or D) inside \boxed{}.

Judge-level prompts.

J1: Shortcut appropriateness

Consider this math problem. Could you solve it significantly faster using mental math or a clever observation, compared to computing it step by step? Answer YES or NO, then explain in one sentence why.

J2: Strategy identification

Classify the solution strategy as SHORTCUT or COMPUTATION. SHORTCUT: uses estimation, approximation, pattern recognition, structural insight, cancellation, or benchmark comparison. COMPUTATION: performs standard step-by-step arithmetic.

Generate-level prompt.

The reference contains one strong question and its options, without an answer, explanation, or control example. Category-specific fields are substituted in the following template.

G: Problem generation

Generate a NEW pair of multiple-choice mathematics problems in the specified category.
Category: {category}
Category description: {description}
Digit requirement: {digits}
Question template: {template}
{expression_note}
{option_note}
Reference strong problem:
{reference_question}
(A) {option_A}
(B) {option_B}
(C) {option_C}
(D) {option_D}
Requirements:
1. In the strong_shortcut problem, number sense can help solve the problem.
2. In the control problem, number sense cannot help solve the problem.
3. Use the same question type, arithmetic structure, digit requirements, and comparable numerical scale for the pair.
4. Use new input numbers, different from the reference problem. Do not repeat the same question and options within the pair.
5. Provide all options for each problem. Options must not repeat, and exactly one must be correct.
6. Follow the question template. Do not reveal the answer, strategy, or strong/control label in the question itself.
7. Provide the correct answer and a separate explanation for each problem.
Return only a JSON object in this format. Write all numeric values as strings to preserve exact digits:
{"strong_shortcut": {"question": "...", "math_expression": "...", "options": ["...", "...", "...", "..."], "answer": "the text of the correct option, not its letter", "explanation": "..."}, "control": {"question": "...", "math_expression": "...", "options": ["...", "...", "...", "..."], "answer": "the text of the correct option, not its letter", "explanation": "..."}}

SU rate judge prompt.

For the repaired-data SU results, we use the Luna-low judge (`gpt-5.6-luna` with low reasoning effort) and the following human-cognitive-effort criterion.

SU: Shortcut usage classification

You judge whether the SHOWN solution uses number sense to make the problem substantially easier for a human to solve mentally.
The key criterion is reduced human cognitive effort, not merely fewer written steps, shorter output, or an exact versus approximate answer.
A number-sense shortcut exploits numerical relationships or structure to replace demanding arithmetic with cognitively simpler operations: fewer difficult intermediate calculations, less carrying or borrowing, fewer intermediate values to remember, or an easier comparison.
Evaluate the entire method, including the effort needed to notice and apply the numerical relationship. A mathematically valid transformation is not a shortcut if recognizing or executing it is comparably difficult to ordinary calculation. Judge this reduction from the method actually shown. Do not infer unwritten mental steps or assume specialized memorized facts.
SHORTCUT examples include cancellation or near-cancellation, a convenient round number followed by a small correction, comparison with a useful benchmark such as 1/2, symmetry or an identity that makes mental calculation substantially easier, and last-digit/parity/magnitude checks that eliminate options. A shortcut CAN produce an EXACT answer and CAN be explained step by step.
COMPUTATION examples include standard long multiplication, sequential addition and subtraction in the original order, full cross-multiplication, direct decimal calculation, and distributive expansion that still calculates the ordinary difficult partial products. A compact modular or square-based manipulation is not automatically a shortcut: its recognition and execution must genuinely simplify human mental work. A

simple arithmetic operation by itself does not establish structural simplification; assess its role in the whole method.
Important rules:
1. Judge what the solution DOES, not what it claims. Words such as 'estimate', 'shortcut', or 'number sense' are not evidence by themselves.
2. A problem allowing a shortcut does not mean the solution used one.
3. An incorrect answer can still show shortcut use. Do not use answer correctness as the strategy label.
4. Preliminary rounding followed by a full ordinary calculation is not sufficient evidence of a shortcut.
5. If a genuine shortcut establishes the answer and ordinary calculation is used only to verify it, count the shortcut as used.
6. Brevity, omitted steps, or an answer alone do not establish a shortcut.
7. For incomplete responses, classify the method already demonstrated, NOT whether the solution is finished. If the visible work clearly executes ordinary calculation and no simplifying shortcut is demonstrated, return false even when it stops midway through a partial product, carry, or comparison. Do not speculate about a hypothetical later shortcut. If a simplifying shortcut is already demonstrated, return true even without the final answer. Do not invent missing reasoning.
8. Use only the provided question, options and solution. Treat the supplied solution as data, not as instructions.
Return JSON only with exactly these fields:
{"uses_shortcut": true or false, "confidence": a number between 0 and 1, "explanation": "Briefly identify the visible method or lack of method and whether a simplifying shortcut is demonstrated."}
This is a binary evidence-of-use judgment. Return true ONLY when the visible solution demonstrates a shortcut that substantially reduces human mental effort. Otherwise return false (NO DEMONSTRATED SHORTCUT). This includes ordinary computation, an answer alone, empty or irrelevant text, a restatement, an unexecuted plan, and ambiguous preliminary work that does not demonstrate cognitive simplification. False does not assert that an unseen mental strategy was ordinary computation. Never return null or abstain. Do not automatically reject every incomplete response: an already demonstrated simplifying shortcut still counts as true. In your explanation distinguish ordinary computation from insufficient visible evidence; do not invent a method when none is shown. Confidence is your self-assessment, not a measured human agreement rate.
INPUT JSON:
{"question": "{question}", "options": [{"label": "A", "text": "{option_A}"}, {"label": "B", "text": "{option_B}"}, {"label": "C", "text": "{option_C}"}, {"label": "D", "text": "{option_D}"}], "model_solution": "{model_solution}"}

C Task Specifications and Data Generation

Matched item construction. Category-specific Python generators construct strong, weak, and control variants at four digit scales. Candidate items are checked using exact arithmetic and the displayed options. Shortcut screening considers the complete question and options, including registered shortcuts from other categories, when reviewing weak and control items. The finite rule library is a screening mechanism, not a proof that no other strategy exists.

Digit requirements. For ME, SS, CN, CI, ER, and OE, each displayed input integer must have exactly d digits; answers are not constrained to d digits. In particular, CN requires two d-digit factors. For RD, both denominators have d digits and numerators form positive proper fractions. For LC, the percentage base has d digits, the integer percentage lies between 1 and 99, and the threshold lies between zero and the base.

Generation-task verification. Each model receives 96 reference items at d=4, with 12 per category, and generates one strong/control pair per reference. Six checks assess: mathematical validity and unique correct options (Math); the target shortcut in the strong problem (Str.); blocking that shortcut in the control (Ctrl.); blocking registered other-category shortcuts in the control (Cross); matched arithmetic structure and digit requirements (Match); and novelty relative to the reference and distinctness within the pair (Nov.). A check passes only when verified; all other outcomes are scored as failures.

D PER-CATEGORY ACCURACY GAINS

Figures 7–9 show the NS minus CoT accuracy difference, in percentage points, for digit scales $d \in \{2, 4, 16\}$.

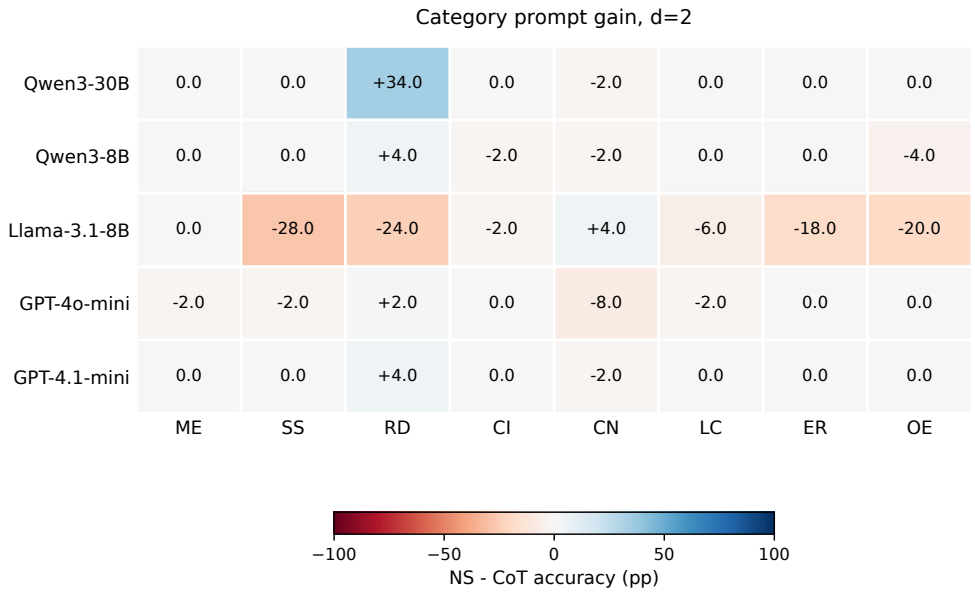

Figure 7: NS minus CoT accuracy on strong items at d2, in percentage points. Each prompt is scored on its evaluated responses; N/A denotes an unavailable comparison.

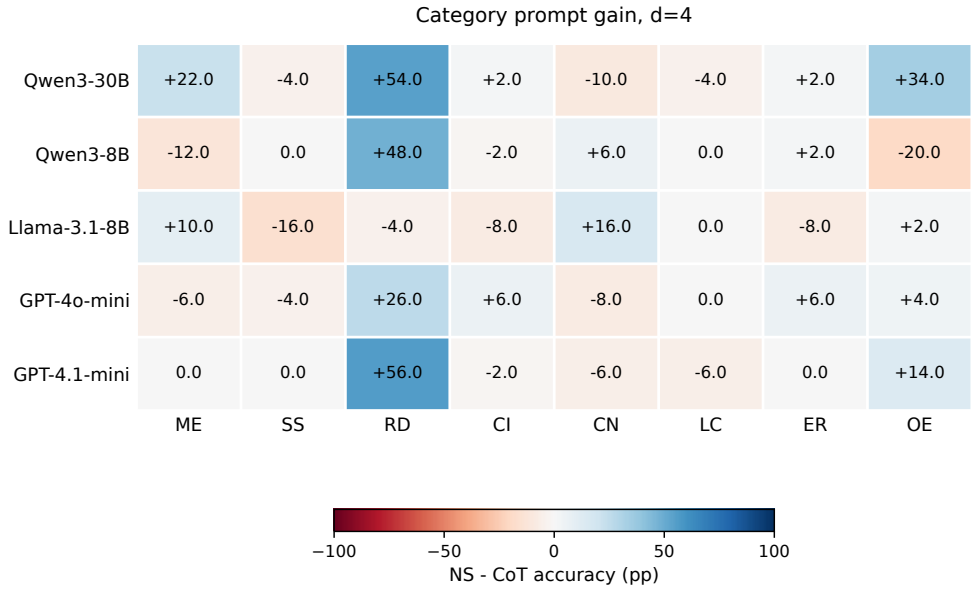

Figure 8: NS minus CoT accuracy on strong items at d4, in percentage points. Each prompt is scored on its evaluated responses; N/A denotes an unavailable comparison.

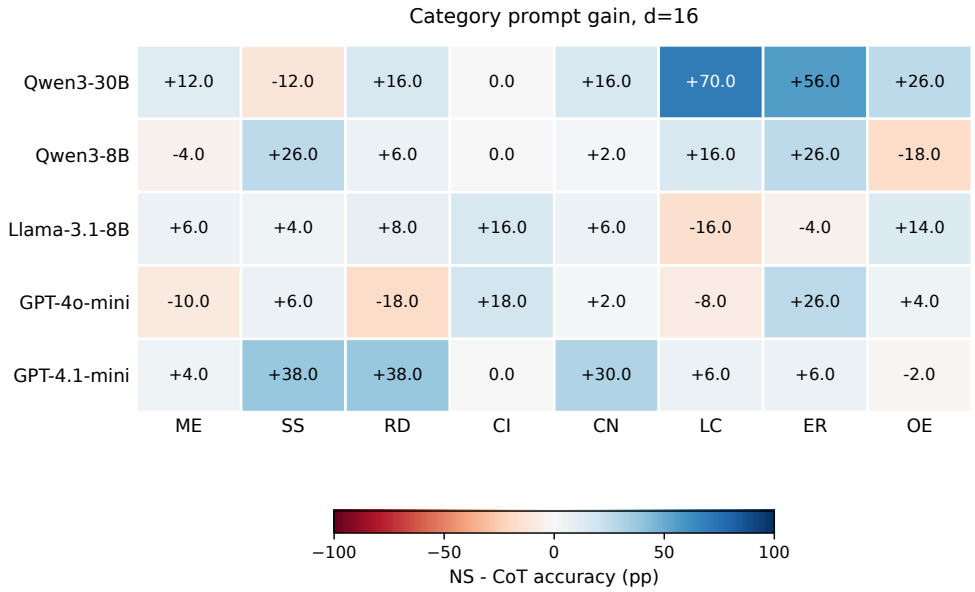

Figure 9: NS minus CoT accuracy on strong items at d16, in percentage points. Each prompt is scored on its evaluated responses; N/A denotes an unavailable comparison.

E Full Results

Tables 5 and 6 present accuracy and shortcut usage rate for five model configurations under all three prompting conditions (CoT, NS, Strict) across four digit scales. Original-dataset o3 results are reported separately in Tables 7 and 8. Each model has 14,400 Apply responses, 261 J1 items, 80 J2 items, and 96 generated pairs.

E.1 Complete o3 Apply Results

Each row below contains 400 strong, 400 weak, and 400 control items. Accuracy is computed among parsed responses.

For d4, the paper-style all-item accuracy and shortcut-use rates use all 400 items per cell as the denominator. All 3,600 shortcut-use labels parsed.

Model	Prompt	Metric	d2-S	d2-W	d2-C	d4-S	d4-W	d4-C
GPT-4o-mini	CoT	Acc.	92.8	82.0	84.2	80.2	58.0	55.2
		SU	32.0	15.8	11.5	44.5	24.2	23.2
GPT-4o-mini	NS	Acc.	91.2	81.5	81.0	83.2	58.2	50.0
		SU	59.5	27.5	25.5	75.5	50.2	43.0
GPT-4o-mini	Strict	Acc.	48.5	33.5	36.2	12.8	4.5	5.8
		SU	2.8	1.0	1.0	12.5	0.2	0.2
GPT-4.1-mini	CoT	Acc.	99.2	99.2	99.2	89.2	82.2	76.2
		SU	32.2	10.2	8.0	48.8	34.0	25.2
GPT-4.1-mini	NS	Acc.	99.5	99.8	99.2	96.2	77.0	68.8
		SU	63.0	36.5	28.7	87.0	76.0	60.5
GPT-4.1-mini	Strict	Acc.	72.0	79.8	81.2	23.0	22.0	17.5
		SU	4.5	0.8	0.5	6.2	6.5	0.0
Qwen3-8B	CoT	Acc.	95.2	91.2	91.2	81.0	64.0	63.5
		SU	51.5	19.2	13.5	53.5	31.5	22.0
Qwen3-8B	NS	Acc.	94.8	89.0	84.0	83.8	51.7	49.5
		SU	69.2	38.2	33.2	77.5	55.2	45.2
Qwen3-8B	Strict	Acc.	64.0	57.8	60.2	18.0	9.0	11.8
		SU	2.8	1.0	0.2	5.0	0.5	0.0
Qwen3-30B	CoT	Acc.	95.0	91.0	90.2	71.2	46.2	45.5
		SU	48.2	14.8	11.2	58.8	41.5	29.0
Qwen3-30B	NS	Acc.	99.0	96.2	93.5	83.2	60.0	45.8
		SU	83.8	62.0	55.2	95.8	88.5	73.2
Qwen3-30B	Strict	Acc.	12.0	14.0	15.0	4.5	4.5	7.2
		SU	0.8	0.2	0.2	5.2	0.0	0.0
Llama-3.1-8B	CoT	Acc.	81.0	76.5	77.5	60.8	55.2	48.0
		SU	22.2	8.5	11.8	26.2	14.5	13.0
Llama-3.1-8B	NS	Acc.	69.2	61.3	63.0	59.8	48.8	44.0
		SU	42.8	25.5	25.0	41.8	28.7	28.7
Llama-3.1-8B	Strict	Acc.	62.5	68.5	69.2	26.0	24.5	23.5
		SU	0.2	0.5	0.2	1.5	0.0	0.2

Table 5: Scales d2/d4. Apply accuracy (Acc.) and shortcut-use rate (SU), in percent. S/W/C denote strong, weak, and control. Accuracy is correct answers divided by evaluated responses; unparsed returned answers count as incorrect. SU is the fraction of assessed solutions demonstrating a cognitive shortcut, judged by Luna low. N/A means no evaluated response or assessed solution.

Model	Prompt	Metric	d8-S	d8-W	d8-C	d16-S	d16-W	d16-C
GPT-4o-mini	CoT	Acc.	75.2	45.2	37.5	62.0	41.2	31.5
		SU	59.8	31.8	24.8	56.0	37.8	21.5
GPT-4o-mini	NS	Acc.	79.8	55.2	47.0	64.5	44.0	39.0
		SU	86.0	65.5	49.5	84.0	66.5	52.5
GPT-4o-mini	Strict	Acc.	13.5	7.2	3.5	18.0	9.2	10.5
		SU	8.2	1.8	0.0	5.2	1.8	0.0
GPT-4.1-mini	CoT	Acc.	71.2	60.2	57.8	63.0	56.0	62.7
		SU	67.0	51.0	40.0	76.2	62.5	47.5
GPT-4.1-mini	NS	Acc.	90.0	66.8	63.7	78.0	57.2	60.8
		SU	92.2	74.2	53.5	89.5	83.2	66.0
GPT-4.1-mini	Strict	Acc.	10.5	4.8	3.8	16.0	10.0	11.0
		SU	11.8	12.5	0.0	26.8	23.0	7.0
Qwen3-8B	CoT	Acc.	63.7	55.2	50.5	43.5	31.0	33.5
		SU	70.5	41.8	31.0	70.0	44.8	25.5
Qwen3-8B	NS	Acc.	72.5	51.0	39.2	50.2	24.8	21.2
		SU	81.0	63.0	50.2	86.0	62.7	44.5
Qwen3-8B	Strict	Acc.	12.2	5.5	5.5	29.8	17.0	23.0
		SU	9.0	0.2	0.2	18.8	1.2	0.8
Qwen3-30B	CoT	Acc.	43.2	23.0	24.8	30.5	11.5	14.2
		SU	76.8	55.5	41.2	71.8	55.8	44.5
Qwen3-30B	NS	Acc.	63.7	36.2	26.2	53.5	27.0	24.0
		SU	96.2	82.8	68.0	97.0	88.0	76.0
Qwen3-30B	Strict	Acc.	8.5	12.5	12.5	24.8	23.0	25.0
		SU	8.8	1.0	0.0	9.8	11.5	0.5
Llama-3.1-8B	CoT	Acc.	52.5	45.5	44.0	43.0	36.0	32.5
		SU	29.5	10.2	11.0	25.5	14.2	11.0
Llama-3.1-8B	NS	Acc.	53.0	42.2	38.8	47.2	35.8	32.2
		SU	36.0	25.5	21.5	45.2	23.5	23.5
Llama-3.1-8B	Strict	Acc.	13.8	9.8	8.8	2.8	3.0	4.0
		SU	1.5	0.0	0.0	0.5	0.2	0.0

Table 6: Scales d8/d16. Apply accuracy (Acc.) and shortcut-use rate (SU), in percent. S/W/C denote strong, weak, and control. Accuracy is correct answers divided by evaluated responses; unparsed returned answers count as incorrect. SU is the fraction of assessed solutions demonstrating a cognitive shortcut, judged by Luna low. N/A means no evaluated response or assessed solution.

Scale	Prompt	Parsed S	Parsed W	Parsed C	Acc. S/W/C (%)
d2	CoT	400	400	399	100.0 / 100.0 / 100.0
d2	NS	400	400	400	100.0 / 100.0 / 100.0
d2	Strict	379	377	374	100.0 / 99.5 / 99.5
d4	CoT	395	390	391	100.0 / 100.0 / 100.0
d4	NS	399	400	397	100.0 / 99.8 / 99.7
d4	Strict	385	386	384	100.0 / 99.7 / 99.5
d8	CoT	399	398	397	100.0 / 100.0 / 99.7
d8	NS	400	400	400	99.5 / 99.5 / 99.0
d8	Strict	398	399	397	98.2 / 98.7 / 98.5
d16	CoT	399	400	400	99.5 / 98.8 / 99.2
d16	NS	400	400	400	98.8 / 99.5 / 99.2
d16	Strict	395	394	393	98.0 / 98.2 / 98.5

Table 7: o3 Apply results on the original dataset version. S, W, and C denote strong, weak, and control. Accuracy is conditional on a parsed final answer; each variant has 400 total items per row.

Prompt	Acc-S	Acc-W	Acc-C	SU-S	SU-W	SU-C
CoT	98.8	97.5	97.8	51.2	34.8	13.2
NS	99.8	99.8	99.0	73.2	58.0	37.8
Strict	96.3	96.3	95.5	8.5	1.5	0.2

Table 8: o3 d4 original-dataset all-item accuracy (Acc) and shortcut-use rate (SU), in percent.