Synthetic Data and the Rise of Spiky Intelligence

Anonymous Authors[1]

Abstract

Synthetic data is increasingly used to train language models, but standard decontamination methods mainly detect duplicated or near-duplicated test examples. They miss a subtler failure mode: *benchmaxxing*, where training data is synthetic and benchmark-adjacent without containing the test set. We study benchmaxxing under controlled conditions and compare it with direct test-set contamination across model scales. We introduce the `Inflation Score`, which measures how much augmentation increases the benchmark–held-out accuracy gap relative to a no-augmentation baseline, using a capability-matched held-out set. We find that benchmark-adjacent synthetic data can inflate benchmark scores as much as direct test-set training. To test transfer, we introduce `Factory-GSM`, a GSM8K rewrite that preserves solution structure while changing surface context. Benchmaxxed models degrade substantially on `Factory-GSM`, suggesting benchmark-specific shortcuts rather than robust reasoning. Gradient similarity detects this specialization better than embedding-based decontamination, though imperfectly. A real-world audit of 46 open-weight models shows similar patterns. Overall, reliable evaluation must detect concentrated benchmark-adjacent training signals, not just copied test examples.

1. Introduction

Current decontamination practice removes training examples that duplicate or near-duplicate benchmark test examples. This protocol was designed for web-scraped corpora, where contamination often appears as accidental copies, and was later extended with paraphrase detection and n-gram filters (Magar & Schwartz, 2022; Sainz et al., 2023; Oren et al., 2023). It is less suited to synthetic-data pipelines, where examples can be generated from public tasks, benchmark-like formats, or benchmark-derived seeds. Such data may concentrate near a benchmark distribution without containing any test example, allowing benchmark scores to rise without a corresponding increase in general capability.

We study this benchmark-local specialization, which we refer to as *spiky intelligence*: performance improves where synthetic training density is high and weakens away from it. This effect is closely related to *benchmaxxing* (Karpathy, 2025)(§2). Prior work on *soft contamination* (Spiesberger et al., 2026) shows that semantic duplicates and variants of evaluation items can evade lexical decontamination. Our focus is different: synthetic data need not duplicate, paraphrase, or semantically match any test item to inflate a benchmark. It can do so by concentrating training signal near the benchmark distribution.

We measure this effect on `GSM8K` (Cobbe et al., 2021). We pretrain 1B, 3B, and 8B models from scratch on 180B tokens of C4, which predates `GSM8K`, and then continually pretrain them under a fixed token budget while varying only the density of synthetic augmentations around one localized training region. Using gradient features, we assign each test example to a training cluster and compare the benchmark-adjacent subset with a difficulty-matched held-out subset. General reasoning gains should improve both subsets; benchmark-local specialization should improve the adjacent subset more.

We summarize this divergence with the `Inflation Score` (§2.2), which normalizes the benchmark–held-out accuracy gap by the no-augmentation baseline gap. Inflation increases with augmentation density and model scale, from 1.6 at 1B to 2.5 at 8B. To test transfer, we introduce `Factory-GSM`, a paired rewrite that changes surface context while preserving numbers, operations, and answers. Inflation reaches 3.1 at 8B on `Factory-GSM`, indicating that the gains depend on benchmark-specific features rather than robust reasoning alone.

Finally, direct training on the `GSM8K` test set produces similar aggregate inflation to the densest synthetic condition, but inflates largely different examples. Thus benchmark-adjacent synthetic data can mimic leakage at the score level while leaving a distinct per-example footprint. Standard

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by *The Impact of Memorization on Trustworthy Foundation Models* Workshop @ ICML. Do not distribute.

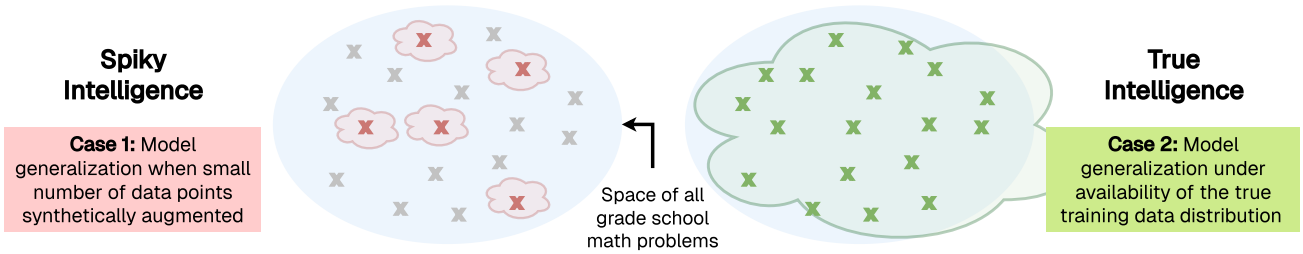

Figure 1. **Spiky intelligence.** Synthetic augmentation produces sharp gains in a local region of the task space (the spike) that transfer poorly to nearby capability-matched problems; broader task coverage produces smoother, more uniformly transferable performance.

input-space decontamination misses this signal; gradient-similarity attribution detects more of it, but remains incomplete. We find similar fingerprints in public open-weight models.

2. Controlled Contamination

We investigate whether synthetic augmentation improves mathematical reasoning broadly or merely concentrates gains around the target region (§2.2); a model that fails to generalize across local regions within a single benchmark is unlikely to generalize across tasks. We define contamination by its effect on *generalization*

Benchmaxxing. Let t denote a target evaluation task, and T a task family intended to measure the same underlying capability, such that $t \subset T$. A training dataset D is *benchmaxxed* with respect to (t, T) if training on D improves performance on t without yielding comparable improvements on held-out tasks drawn from T.

This definition is not limited to verbatim duplication: synthetic rephrasings, alternate solution paths, and examples sharing the same local reasoning structure can all produce spiky intelligence if they inflate performance on the target task without comparable transfer to the family.

2.1. Choice of Datasets and Models

We use `GSM8K` because it has public synthetic augmentations and a clean historical control. Our 1B, 3B, and 8B base models are pretrained from scratch on 180B tokens of C4 (Raffel et al., 2020), which predates `GSM8K`. We then run 2B-token continual pretraining (CPT) from each base model with a fixed mixture of `GSM8K`-derived task data, MATH-derived auxiliary data, and instruction-following data (Appendix F). Instruction data is always half the CPT mixture; across conditions, we vary only the target-cluster `OpenMathInstruct-2` density.

2.2. Experimental setup

The best way to study this would be to hold out a fresh, unreleased test split and ask whether benchmark gains transfer there. Since `GSM8K` has no such split, we create the same test inside the benchmark: we cluster the `GSM8K` train set using answer-conditioned gradient features and choose one cluster, A, as the target region. During CPT, only examples in A receive synthetic `OpenMathInstruct-2` augmentations.

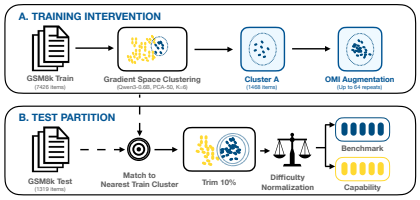

Figure 2. **Training intervention.** Train examples are clustered in gradient space; one cluster, A, receives synthetic augmentation. Test examples are routed to their nearest training cluster, yielding a benchmark subset adjacent to A and a difficulty-matched capability subset drawn from the rest.

Transfer test. Each test item is routed to its nearest training cluster. Items adjacent to A form the *benchmark* subset; difficulty-matched items routed elsewhere form the *capability* subset (Figure 2). If augmentation teaches general mathematical reasoning, both subsets should improve together. If it induces benchmark-local specialization, gains should concentrate on the benchmark subset. Appendix D gives the clustering, trimming, and matching details.

Metric. Synthetic density $r \in \{1, 2, 4, 8, 16, 32, 64\}$ is the number of synthetic solutions per target question. Let $s_{\text{bench}}(r)$ and $s_{\text{cap}}(r)$ be accuracies after CPT, and let $g(r) = s_{\text{bench}}(r) - s_{\text{cap}}(r)$. We report

$$\texttt{Inflation Score}(r) = \frac{g(r)}{g(1)}$$

`Inflation Score` $>$ 1 means the benchmark–capability gap widened; values near 1 mean transfer is broad.

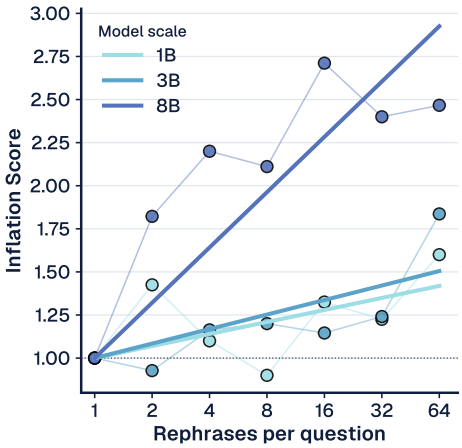

Figure 3. **Synthetic density concentrates gains on benchmark-adjacent items, more so at larger scale.** `Inflation Score` compares benchmark-adjacent gains to capability-subset gains, normalized to the single-augmentation baseline. Absolute accuracy curves are in Appendix Fig. 5.

3. Experimental results

3.1. Synthetic density inflates local benchmark performance

We ask whether increasing synthetic density produces broad reasoning transfer or only benchmark-local gains. We track how the `Inflation Score` changes as the number of synthetic solutions per target question increases.

Higher synthetic density concentrates gains near the augmented region. At low densities, benchmark and capability performance improve together, and the intervention behaves like ordinary generalization. As density increases, however, the benchmark subset gains accuracy faster than the capability subset, so the gap between them widens with density even though both subsets continue to improve in absolute terms. We hypothesize that the model exhausts the generalizable reasoning structures available in the augmentation and begins overfitting to the narrow reasoning paths and stylistic artifacts of the target cluster.

The aggregate benchmark score can rise even as the gain becomes less transferable. This phenomenon is invisible when evaluating on standard aggregate held-out sets, because the aggregate score averages the faster-rising benchmark subset with the slower-rising capability subset and overstates how much broad capability has improved; the `Inflation Score` makes the gap visible by isolating the performance ratio between the augmented neighborhood and the rest of the test set.

Inflation increases with model size. A common assumption is that larger models rely less on narrow memorization and more on generalizable structure, so the inflation effect should shrink with scale. At the highest augmentation density, the `Inflation Score` reaches roughly 1.6 for the 1B, 1.8 for the 3B, and 2.5 for the 8B (Fig. 3). The 1B and 3B curves are noisier, while the 8B curves rise more cleanly. Our results indicate that scale does not protect against local inflation, and if anything larger models show a greater benchmark–capability gap.

> **Takeaway:** The benchmark-capability gap widens monotonically with density at all three scales.

3.2. Do benchmark-local gains transfer under rewrites?

The capability split in §3.1 stays within the `GSM8K` test distribution, so we add a stricter transfer test: `Factory-GSM`, a manufacturing-domain rewrite of `GSM8K`.

Constructing `Factory-GSM`. We rewrite each test question into an industrial setting, changing names, objects, and context while keeping numerical values, operations, solution structure, and final answer fixed. No rewritten item appears in training, and each item inherits the benchmark/capability label of its `GSM8K` source; Appendix J gives generation and verification details.

Simple rewrites weaken transfer. If dense synthetic augmentation taught the underlying arithmetic procedures, gains should transfer to these rewrites; they do not. At 64× augmentation, the rewrite-based `Inflation Score` is 1.94, 2.42, and 3.11 for the 1B, 3B, and 8B models, larger than the within-`GSM8K` scores of 1.60, 1.84, and 2.47 from §3.1 (Appendix Fig. 6). The surface rewrite strengthens the inflation signal even though the computation is unchanged.

Scale amplifies surface dependence. The transfer gap grows with model size: larger models gain more on the original benchmark subset, but those gains transfer less reliably to `Factory-GSM`. Dense benchmark-adjacent augmentation makes larger models more sensitive to `GSM8K` surface form, not more robust to it.

Appendix I gives complementary evidence: OMI rephrases stay near their parents in embedding space (98% in same cluster) but not in gradient space (40%), so topical similarity does not imply training-dynamics similarity.

> **Takeaway.** Dense synthetic augmentation teaches arithmetic procedures, but also benchmark-specific surface regularities. `Factory-GSM` highlights this by changing context while preserving solution structure.

4. Synthetic Inflation Matches Leakage in Aggregate but Not in Mechanism

A natural question is how dense synthetic augmentation compares to direct evaluation leakage. We pair the 64× OMI baseline of §2 with an explicit-leakage condition that inserts `GSM8K`-test examples into the CPT mixture, then convert the trajectory into effective evaluation-set repeats.

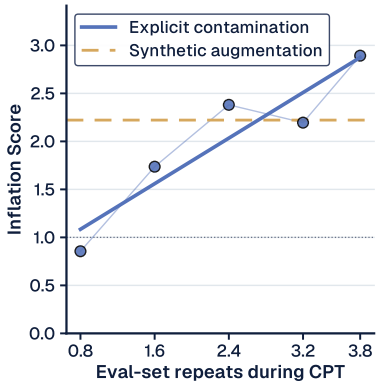

Figure 4. **Dense synthetic augmentation matches a few evaluation-set repeats.** `Inflation Score` at 8B under progressive test-set injection; the dashed line marks the 64× `OpenMathInstruct` reference.

Its `Inflation Score` crosses the synthetic baseline at roughly 2.2 effective copies of the test set (Figure 4; setup details in Appendix H).

Aggregate inflation is silent on mechanism. At matched `Inflation Score`, the newly correct items overlap only weakly (Appendix Fig. 7): explicit injection improves exposed test items, while synthetic augmentation improves benchmark-adjacent items. Input-space decontamination is built to catch the first route, so it can miss the second while certifying a corpus as clean; this motivates the public-model audit in §6.

5. Limitations of Current Decontamination Approaches

Existing decontamination methods look for input-space overlap: n-grams, embedding retrieval, or LLM paraphrase checks (Brown et al., 2020; Touvron et al., 2023; OpenAI, 2023; Abdin et al., 2024b; Lee et al., 2022; Yang et al., 2023; Toshniwal et al., 2024; Ravaut et al., 2024; Sainz et al., 2023). These methods catch copies and paraphrases, but benchmaxxing need not look like copying. It can arise whenever training and evaluation examples induce similar parameter updates.

We compare two nearest-neighbor signals, embedding similarity and gradient similarity, using ROC-AUC to predict which evaluation items become newly correct after synthetic augmentation.

Semantic embeddings carry essentially no signal. Input-space proximity is uninformative: nearest-neighbor cosine similarity with Qwen3-Embedding-0.6B (Zhang et al., 2025) achieves 0.54 AUC, near chance (Appendix Fig. 8, left). The intuition is simple: embeddings group by topic, while learning follows solution updates. Two similar stories can require different operations; two unrelated stories can train the same arithmetic pattern.

Gradient space is where the signal lives. Since benchmark inflation is driven by optimization rather than topical overlap, parameter-update proximity is the natural signal. Per-example loss gradients, projected to 8,192 dimensions, raise predictiveness to 0.68 AUC (Appendix Fig. 8).

Small external gradient models work best. Among gradient representations, small external reference models such as Qwen3-0.6B work best (Appendix Fig. 8, right). Larger reference models and self-gradients are less discriminative, plausibly because strong math models have already internalized many solution patterns while self-gradients are weak before CPT. Appendix C gives scope details.

6. Auditing Public Open-Weight Models

The controlled experiments give us a fingerprint of benchmark-local learning. The public-model audit asks whether the same fingerprint appears when training data is not observable. We cannot infer a model's data source from behavior alone, but we can test transfer: robust arithmetic skill should survive transformations that keep the solution structure intact.

We audit 46 open-weight models from 1B to roughly 140B parameters along two axes: *Context Gap*, the drop from `GSM8K` to `Factory-GSM`; and *Numerical Gap*, the drop to `GSM-Symbolic` (Mirzadeh et al., 2024). The two transformations preserve the operation graph while changing either the story or the numbers.

Family signatures emerge (Appendix Fig. 9). Qwen-Math and Qwen-Instruct models (Yang et al., 2024; 2025b;a) lose 5–14pp under context rewrites, while OLMo-2 (OLMo Team, 2024) has small Context Gaps but 7–11 point Numerical Gaps. This mirrors our controlled diverse-rephrase condition: exposure to fixed numerical instances can improve wording invariance without improving numerical invariance.

These gaps are diagnostic rather than forensic: they show what property a benchmark score depends on, not what a model trained on. Appendix K gives the protocol and full table.

7. Conclusion

Dense benchmark-adjacent synthetic data can produce *spiky intelligence*: gains concentrate near the benchmark, grow with density and scale, weaken under `Factory-GSM` transfer, and can match explicit leakage while improving different examples. A benchmark score should therefore be read as local until it survives transfer tests. Input-space decontamination misses this failure mode, gradient similarity helps but is incomplete, and public-model audits show similar sensitivities; reliable evaluation needs transformations that preserve the skill while moving away from the benchmark surface. We discuss limitations in Appendix L.

References

Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024a. URL https://arxiv.org/abs/2404.14219.

Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., Harrison, M., Hewett, R. J., Javaheripi, M., Kauffmann, P., Lee, J. R., Lee, Y. T., Li, Y., Liu, W., Mendes, C. C. T., Nguyen, A., Price, E., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Wang, X., Ward, R., Wu, Y., Yu, D., Zhang, C., and Zhang, Y. Phi-4 technical report, 2024b. URL https://arxiv.org/abs/2412.08905.

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics, 2023. URL https://arxiv.org/abs/2310.10631.

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling. In *ICML*, 2023. URL https://arxiv.org/abs/2304.01373.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020. URL https://arxiv.org/abs/2005.14165.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., and Raffel, C. Extracting training data from large language models. In *USENIX Security*, 2021. URL https://arxiv.org/abs/2012.07805.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramèr, F., and Zhang, C. Quantifying memorization across neural language models. In *ICLR*, 2023.

Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. In *ICALP*, 2002.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Dodge, J., Sap, M., Marasovic, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M., and Gardner, M. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021. URL https://arxiv.org/abs/2104.08758.

Elazar, Y., Bhagia, A., Magnusson, I., Ravichander, A., Schwenk, D., Suhr, A., Walsh, P., Groeneveld, D., Soldaini, L., Singh, S., Hajishirzi, H., Smith, N. A., and Dodge, J. What's in my big data? In *ICLR*, 2024. URL https://arxiv.org/abs/2310.20707.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800GB dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

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

Golchin, S. and Surdeanu, M. Data contamination quiz: A tool to detect and estimate contamination in large language models, 2023a. URL https://arxiv.org/abs/2311.06233.

Golchin, S. and Surdeanu, M. Time travel in LLMs: Tracing data contamination in large language models, 2023b. URL https://arxiv.org/abs/2308.08493.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A. H., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K. R., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Khot, T., Merrill, W., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Strubell, E., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Zettlemoyer, L., Dodge, J., Lo, K., Soldaini, L., Smith, N. A., and Hajishirzi, H. OLMo: Accelerating the science of language models, 2024. URL https://arxiv.org/abs/2402.00838.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., Hubinger, E., Lukošiūtė, K., Nguyen, K., Joseph, N., McCandlish, S., Kaplan, J., and Bowman, S. R. Studying large language model generalization with influence functions, 2023. URL https://arxiv.org/abs/2308.03296.

Hugging Face Smol Team. SmolLM3: Smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., and El Sayed, W. Mistral 7B, 2023. URL https://arxiv.org/abs/2310.06825.

Kandpal, N., Deng, H., Roberts, A., Wallace, E., and Raffel, C. Large language models struggle to learn long-tail knowledge, 2022. URL https://arxiv.org/abs/2211.08411.

Karpathy, A. 2025 LLM year in review. https://karpathy.bearblog.dev/year-in-review-2025/, Dec 2025.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *ICML*, 2017.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. In *ACL*, 2022.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G., and Misra, V. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Liao, M., Luo, W., Li, C., Wu, J., and Fan, K. MARIO: MAth Reasoning with code Interpreter Output – a reproducible pipeline, 2024. URL https://arxiv.org/abs/2401.08190.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Liu, J., Min, S., Zettlemoyer, L., Choi, Y., and Hajishirzi, H. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens, 2024. URL https://arxiv.org/abs/2401.17377.

Longpre, S., Yauney, G., Reif, E., Lee, K., Roberts, A., Zoph, B., Zhou, D., Wei, J., Robinson, K., Mimno, D., and Ippolito, D. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity, 2023. URL https://arxiv.org/abs/2305.13169.

Magar, I. and Schwartz, R. Data contamination: From memorization to exploitation. In *ACL*, 2022.

Maini, P., Yaghini, M., and Papernot, N. Dataset inference: Ownership resolution in machine learning. In *ICLR*, 2021.

Maini, P., Seto, S., Bai, H., Grangier, D., Zhang, Y., and Jaitly, N. Rephrasing the web: A recipe for compute and data-efficient language modeling, 2024. URL https://arxiv.org/abs/2401.16380.

Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., and Farajtabar, M. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

Mitra, A., Khanpour, H., Rosset, C., and Awadallah, A. Orca-math: Unlocking the potential of SLMs in grade school math, 2024. URL https://arxiv.org/abs/2402.14830.

Muennighoff, N., Rush, A. M., Barak, B., Scao, T. L., Piktus, A., Tazi, N., Pyysalo, S., Wolf, T., and Raffel, C. Scaling data-constrained language models, 2023. URL https://arxiv.org/abs/2305.16264.

NVIDIA. NVIDIA-Nemotron and Llama-Nemotron models. https://huggingface.co/collections/nvidia/nvidia-nemotron-v3, 2025a.

NVIDIA. OpenMath-Nemotron: Open-source math-specialized reasoning models. https://huggingface.co/collections/nvidia/openmath-65c5619de2ba059be0775014, 2025b.

OLMo Team. 2 OLMo 2 furious, 2024. URL https://arxiv.org/abs/2501.00656.

OLMo Team. OLMo 3: Open language models. https://huggingface.co/collections/allenai/olmo-3, 2025.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. gpt-oss: Open-weight reasoning models. https://openai.com/index/introducing-gpt-oss/, 2025.

OpenAI. Why swe-bench verified no longer measures frontier coding capabilities. https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/, February 2026. Accessed: 2026-05-04.

Oren, Y., Meister, N., Chatterji, N. S., Ladhak, F., and Hashimoto, T. B. Proving test set contamination in black box language models, 2023. URL https://arxiv.org/abs/2310.17623.

Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. TRAK: Attributing model behavior at scale. In *ICML*, 2023.

Paster, K., Santos, M. D., Azerbayev, Z., and Ba, J. OpenWebMath: An open dataset of high-quality mathematical web text, 2023. URL https://arxiv.org/abs/2310.06786.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only, 2023. URL https://arxiv.org/abs/2306.01116.

Penedo, G., Kydlíček, H., Ben allal, L., Lozhkov, A., Mitchell, M., Raffel, C., Von Werra, L., and Wolf, T. The FineWeb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. In *NeurIPS*, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020.

Ravaut, M., Ding, B., Jiao, F., Chen, H., Li, X., Zhao, R., Qin, C., Xiong, C., and Joty, S. A comprehensive survey of contamination detection methods in large language models, 2024. URL https://arxiv.org/abs/2404.00699.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof Q&A benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Riddell, M., Ni, A., and Cohan, A. Quantifying contamination in evaluating code generation capabilities of language models, 2024. URL https://arxiv.org/abs/2403.04811.

Sainz, O., Campos, J. A., García-Ferrero, I., Etxaniz, J., de Lacalle, O. L., and Agirre, E. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of EMNLP*, 2023.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Shi, W., Ajith, A., Xia, M., Huang, Y., Liu, D., Blevins, T., Chen, D., and Zettlemoyer, L. Detecting pretraining data from large language models. In *ICLR*, 2024.

Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Authur, R., Bogin, B., Chandu, K., Dumas, J., Elazar, Y., Hofmann, V., Jha, A. H., Kumar, S., Lucy, L., Lyu, X., Lambert, N., Magnusson, I., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Ravichander, A., Richardson, K., Shen, Z., Strubell, E., Subramani, N., Tafjord, O., Walsh, P., Zettlemoyer, L., Smith, N. A., Hajishirzi, H., Beltagy, I., Groeneveld, D., Dodge, J., and Lo, K. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024. URL https://arxiv.org/abs/2402.00159.

Spiesberger, A., Vazquez, J. J., Pochinkov, N., Gavenciak, T., Grietzer, P., Leech, G., and Schoots, N. Soft contamination means benchmarks test shallow generalization, 2026. URL https://arxiv.org/abs/2602.12413.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2022. URL https://arxiv.org/abs/2206.04615.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca: A strong, replicable instruction-following model. Stanford Center for Research on Foundation Models. https://crfm.stanford.edu/2023/03/13/alpaca.html, 2023.

Tirumala, K., Markosyan, A., Zettlemoyer, L., and Aghajanyan, A. Memorization without overfitting: Analyzing the training dynamics of large language models. In *NeurIPS*, 2022.

Toshniwal, S., Du, W., Moshkov, I., Kisacanin, B., Ayrapetyan, A., and Gitman, I. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data, 2024. URL https://arxiv.org/abs/2410.01560.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions, 2022. URL https://arxiv.org/abs/2212.10560.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL https://arxiv.org/abs/2201.11903.

Wettig, A., Gupta, A., Malik, S., and Chen, D. QuRating: Selecting high-quality data for training language models, 2024. URL https://arxiv.org/abs/2402.09739.

Xie, S. M., Santurkar, S., Ma, T., and Liang, P. Data selection for language models via importance resampling. In *NeurIPS*, 2023.

Xu, R., Wang, Z., Fan, R.-Z., and Liu, P. Benchmarking benchmark leakage in large language models, 2024. URL https://arxiv.org/abs/2404.18824.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025b. URL https://arxiv.org/abs/2412.15115.

Yang, S., Chiang, W.-L., Zheng, L., Gonzalez, J. E., and Stoica, I. Rethinking benchmark and contamination for language models with rephrased samples, 2023. URL https://arxiv.org/abs/2311.04850.

Ye, J., Liu, P., Sun, T., Zhan, J., Zhou, Y., and Qiu, X. Data mixing laws: Optimizing data mixtures by predicting language modeling performance, 2024. URL https://arxiv.org/abs/2403.16952.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. MetaMath: Bootstrap your own mathematical questions for large language models, 2023. URL https://arxiv.org/abs/2309.12284.

Yue, X., Qu, X., Zhang, G., Fu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. MAmmoTH: Building math generalist models through hybrid instruction tuning, 2023. URL https://arxiv.org/abs/2309.05653.

Zawalski, M., Boubdir, M., Bałazy, K., Nushi, B., and Ribalta, P. Detecting data contamination in LLMs via in-

context learning, 2025. URL https://arxiv.org/abs/2510.27055.

Zhang, H., Da, J., Lee, D., Robinson, V., Wu, C., Song, W., Zhao, T., Raja, P., Slack, D., Zhuang, C., Lyu, Q., Hendryx, S., Kaplan, R., Lunati, M., and Yue, S. A careful examination of large language model performance on grade school arithmetic, 2024. URL https://arxiv.org/abs/2405.00332.

Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., and Zhou, J. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Zhou, K., Zhu, Y., Chen, Z., Chen, W., Zhao, W. X., Chen, X., Lin, Y., Wen, J.-R., and Han, J. Don't make your LLM an evaluation benchmark cheater, 2023. URL https://arxiv.org/abs/2311.01964.

A. Extended Related Work

Benchmark contamination detection. Detecting whether evaluation benchmarks appear in training corpora is a long-standing concern in NLP. Early approaches rely on exact or near-exact string matching: n-gram overlap, character-level edit distance, and MinHash-based deduplication (Brown et al., 2020; Gao et al., 2020; Lee et al., 2022; Dodge et al., 2021). More recent work uses embedding similarity or paraphrase detection to catch semantically rephrased overlaps (Spiesberger et al., 2026; Yang et al., 2023). A complementary line probes for contamination through model behavior: Golchin & Surdeanu (2023b) detect leakage by testing whether models can reconstruct benchmark text verbatim; MIN-K% exploits the observation that memorized text is assigned anomalously high likelihood (Shi et al., 2024); and Oren et al. (2023) develop statistical tests for membership inference on black-box models. Other approaches identify contamination through perturbation sensitivity and option-order dependence (Golchin & Surdeanu, 2023a; Zhou et al., 2023). CoDeC (Zawalski et al., 2025) detects and quantifies contamination by observing how in-context examples affect model confidence, exploiting the finding that ICL boosts confidence on unseen data but can reduce it on memorized training data; they audit roughly 60 public models and release a contamination leaderboard. Large-scale audits have found evidence of leakage across numerous public benchmarks and reasoning tasks (Sainz et al., 2023; Xu et al., 2024; Riddell et al., 2024; Magar & Schwartz, 2022), and surveys provide a comprehensive taxonomy of detection methods (Ravaut et al., 2024). Infini-gram (Liu et al., 2024) enables efficient n-gram lookup at trillion-token scale. Our work differs: rather than detecting input-space overlap, we measure the downstream performance effect of training exposure and compare it against explicit evaluation injection.

Memorization in language models. Closely related to contamination is the question of how models memorize training data. Carlini et al. (2021) show that generative models can be induced to emit verbatim training sequences, and subsequent work quantifies how memorization scales with model capacity, data repetition, and example rarity (Carlini et al., 2023; Tirumala et al., 2022). Deduplication reduces both memorization and privacy risks (Kandpal et al., 2022; Lee et al., 2022). The Pythia suite provides controlled checkpoints for studying memorization dynamics across training (Biderman et al., 2023). Influence functions (Koh & Liang, 2017; Pruthi et al., 2020; Grosse et al., 2023) and scalable attributors such as TRAK (Park et al., 2023) estimate which training examples are responsible for specific predictions. Dataset Inference (Maini et al., 2021) tests whether a dataset was used to train a given model, operating at the dataset level rather than the example level; our gradient-similarity decontamination analysis connects to the same underlying intuition.

Decontamination of pretraining corpora. Model developers typically address contamination through preprocessing: removing training documents whose content overlaps with held-out evaluation items under n-gram, edit-distance, or embedding-based thresholds (Brown et al., 2020; Touvron et al., 2023; OpenAI, 2023; Abdin et al., 2024b). Corpus auditing tools such as "What's In My Big Data?" (Elazar et al., 2024) and Infini-gram (Liu et al., 2024) search for near-duplicates of specified strings at trillion-token scale. Major pretraining corpora, including the Pile (Gao et al., 2020), Dolma (Soldaini et al., 2024), RefinedWeb (Penedo et al., 2023), and Llemma's math corpus (Azerbayev et al., 2023), all include explicit decontamination passes. Longpre et al. (2023) characterize how contamination criteria and data-processing choices affect downstream evaluation performance; Dodge et al. (2021) document the extent of benchmark leakage in the C4 corpus specifically. Despite these efforts, our results show that input-space decontamination is blind to benchmaxxing: synthetic augmentations generated from benchmark training seeds pass all standard filters by construction, while still producing measurable benchmark-local inflation.

Held-out and perturbed evaluations. Several prior works test whether strong benchmark performance reflects genuine generalization. GSM-Symbolic perturbs `GSM8K`-style problems and shows brittle performance under symbolic changes (Mirzadeh et al., 2024). GSM1k constructs a fresh isomorphic benchmark and measures the gap to `GSM8K` performance across public models (Zhang et al., 2024). GPQA constructs graduate-level questions designed to resist web search and memorization (Rein et al., 2023). HumanEval tests code generation on hand-written problems withheld from training (Chen et al., 2021). BIG-Bench assembles hundreds of tasks specifically intended to stay ahead of training data (Srivastava et al., 2022). OpenAI (2026) report that SWE-bench Verified has been saturated by targeted contamination. Our `Factory-GSM` evaluation contributes a held-out variant of `GSM8K` with surface-rewritten questions and new real-world contexts, complementing these eval-side interventions with a training-side causal analysis.

Synthetic math data. Synthetic math datasets have become a primary vehicle for post-training improvement. MetaMath (Yu et al., 2023) rewrites `GSM8K` and MATH problems from multiple perspectives to broaden coverage. MAmmoTH (Yue et al., 2023) combines chain-of-thought and program-of-thought solutions across diverse math datasets.

DeepSeekMath (Shao et al., 2024) curates a web-sourced math corpus and fine-tunes with group relative policy optimization. Orca-Math (Mitra et al., 2024) uses GPT-4-generated solution data to boost small-model math performance. Let's Verify Step by Step (Lightman et al., 2023) introduces process reward models for step-level math supervision. MARIO augments reasoning with code interpreter output (Liao et al., 2024). Minerva (Lewkowycz et al., 2022) and Qwen2.5-Math (Yang et al., 2024) demonstrate the impact of large-scale math-domain pretraining. OpenMathInstruct-2 (Toshniwal et al., 2024) and OpenWebMath (Paster et al., 2023) further scale math-specialized training. Chain-of-thought prompting (Wei et al., 2022) underpins many of these pipelines. These datasets concentrate training signal around specific benchmarks, which is precisely the mechanism whose downstream effects we measure.

Rephrasing the web and data augmentation. A growing literature augments pretraining corpora by rewriting web text to improve quality or diversity. Maini et al. (2024) demonstrate that replacing low-quality web documents with model-rephrased versions substantially improves language modeling efficiency and downstream performance, providing both a recipe and a motivation for rephrase-based augmentation at scale. Self-Instruct (Wang et al., 2022) uses a language model to generate instruction-following data from a small seed set, and Alpaca extends this to fine-tuning from 52k GPT-generated examples (Taori et al., 2023). Yang et al. (2023) show that rephrasing benchmark items, without changing the answers, can mislead evaluations by reintroducing semantically equivalent training signal, directly motivating our study.

Data curation and pretraining corpora. The composition of pretraining corpora has a large effect on downstream capabilities. Muennighoff et al. (2023) show that repeating data hurts less than expected and derive scaling laws for data-constrained regimes. FineWeb (Penedo et al., 2024) applies aggressive quality filtering to Common Crawl to produce a high-quality web corpus. Data Selection via Importance Resampling (DSIR) (Xie et al., 2023) and QuRating (Wettig et al., 2024) select training subsets that align with a target distribution. Ye et al. (2024) identify scaling laws for domain mixture proportions. OLMo (Groeneveld et al., 2024) and Llama 3 (Grattafiori et al., 2024) both describe multi-stage data pipelines that include synthetic augmentation and decontamination; OLMo-2 tops our audit in §6, with documented OMI-2 inclusion. Phi-4 (Abdin et al., 2024b) also relies heavily on synthetic data and represents a prominent example of the pipeline we study.

Our contribution. We provide the first systematic causal comparison of benchmaxxing and explicit evaluation contamination. By augmenting a localized region of `GSM8K` training examples with synthetic rephrases and comparing gains across near and far evaluation slices, we separate broad generalization from benchmark-local inflation. We calibrate this inflation against explicit eval-set injection, test the limits of input-space decontamination at 0.54 AUC (embedding) and 0.68 AUC (gradient similarity), introduce `Factory-GSM` as a surface-rewrite evaluation, and audit 46 open-weight models for the benchmaxxing fingerprint. The Context Gap and Numerical Gap defined in §6 expose family-specific patterns in this fingerprint, with Qwen variants depending heavily on benchmark wording and OLMo-2 variants on the original numbers.

B. Difficulty normalization

The original benchmark and capability subsets differ in baseline difficulty. We estimate baseline difficulty with an 8B reference model trained uniformly on the full `GSM8K` training set. Without any targeting of cluster A, the model scores 32.5% on the benchmark subset and 24.6% on the capability subset, so the benchmark subset is easier by 7.9 percentage points. We construct difficulty-matched subsets by subsampling items until both subsets have the same size and the same reference-model accuracy, computed across three seeds: we remove randomly selected reference-correct items from the benchmark subset and reference-incorrect items from the capability subset. The resulting benchmark* and capability* partitions contain 464 examples each and both have 28.5% accuracy under the reference model. The final `Inflation Score` is computed on this difficulty matched partition.

C. Decontamination Evaluation Scope

The decontamination experiment in §5 evaluates semantic embedding similarity directly and treats lexical methods as a weaker input-space baseline: exact match and n-gram filters flag a strict subset of the surface overlap that embedding retrieval can represent. We therefore use the embedding result to test whether input-space proximity is informative at all. Our conclusion applies to off-the-shelf input-space decontamination; specialized math embeddings or stronger attribution methods could improve on this signal, but would require controlled ground truth of the kind provided by our experiments.

D. Controlled-Contamination Setup Details

We use the GSM8K training set to define the target region. For each of the 7,426 training examples, we compute answer-conditioned gradients with Qwen3-0.6B, reduce them to 8,192 dimensions with count-sketch projection (Charikar et al., 2002), reduce again to 50 dimensions with PCA, and cluster with K-means at $K = 6$. One cluster, A, contains 1,468 examples and is the only region augmented during CPT.

To construct the transfer test, we assign each GSM8K test item to its nearest training example in gradient space. Items assigned to cluster A form the benchmark subset; items assigned elsewhere form the capability subset. We remove ambiguous boundary cases with a 10% relative-margin trim and then normalize difficulty across the two partitions as described in Appendix B. Synthetic density r is the number of OpenMathInstruct-2 solution variants per target question, with $r = 1$ as the single-solution baseline.

E. Additional Main-Experiment Figures

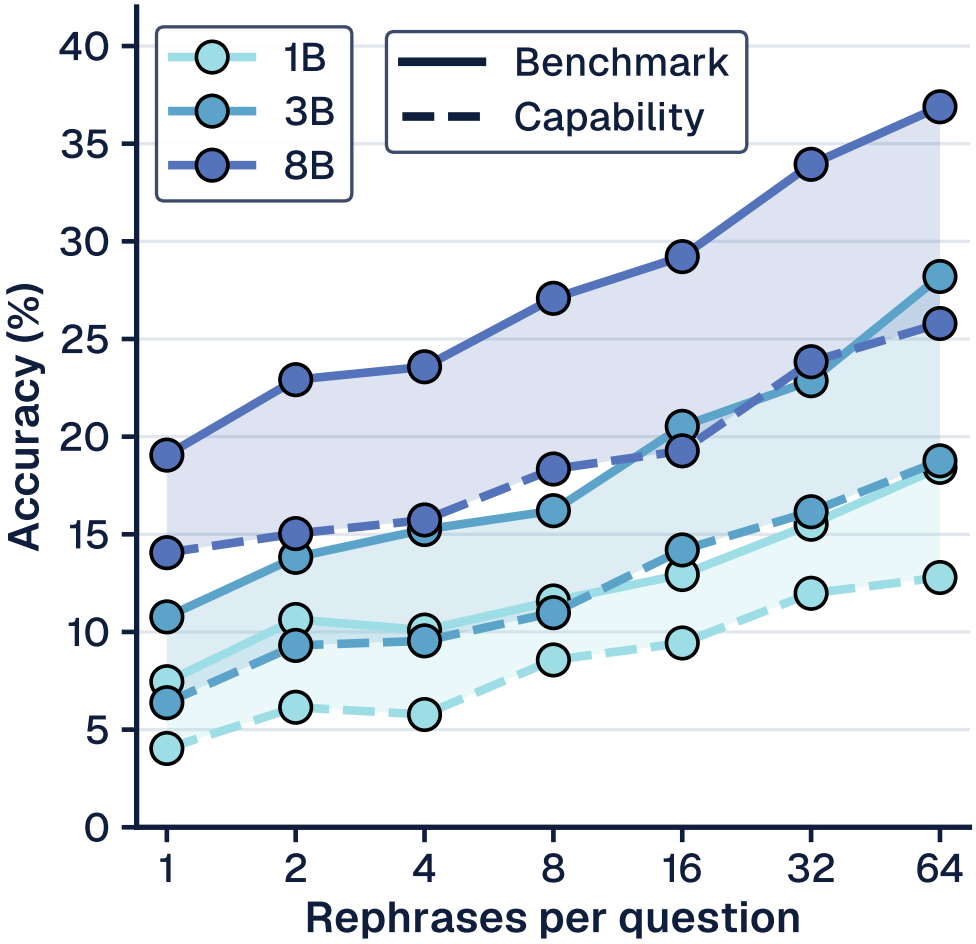

Figure 5. **Benchmark and capability accuracy across synthetic density.** Accuracy curves underlying Figure 3; the benchmark-adjacent subset gains faster than the capability subset at high density.

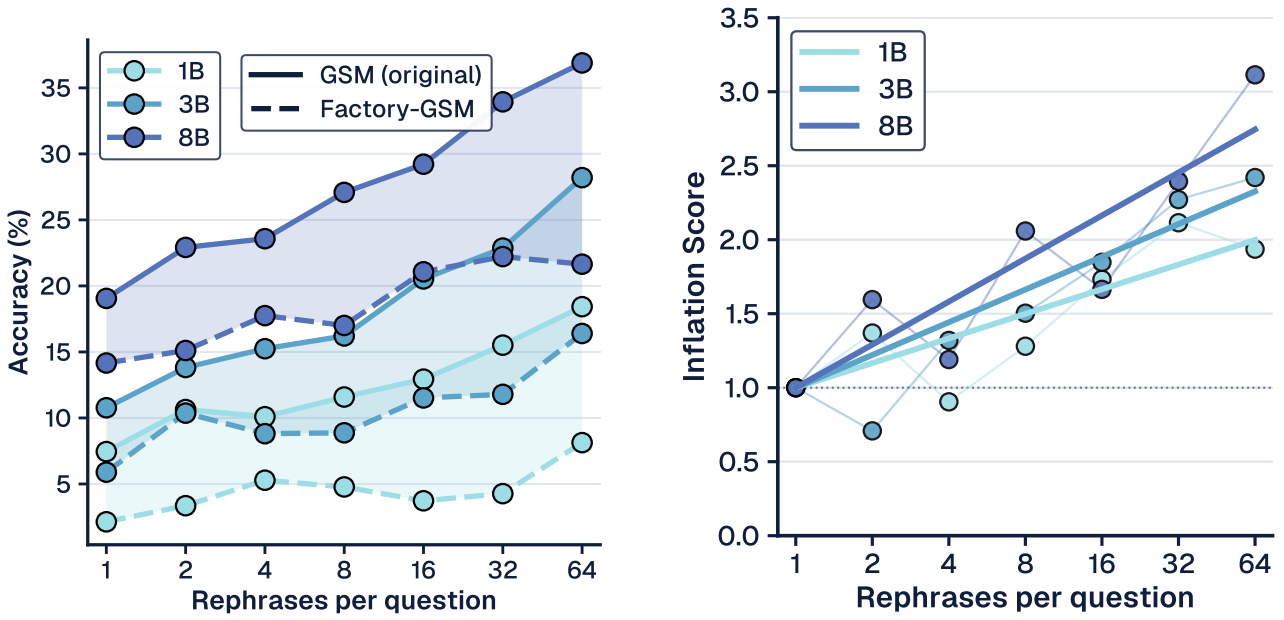

Figure 6. **Surface rewrites widen the benchmark–capability gap, with the effect growing with scale.** *Left*: `GSM`-near and `Factory-GSM`-near accuracy across augmentation density. *Right*: `Inflation Score` comparing gains on `GSM`-near items with gains on their `Factory-GSM` rewrites.

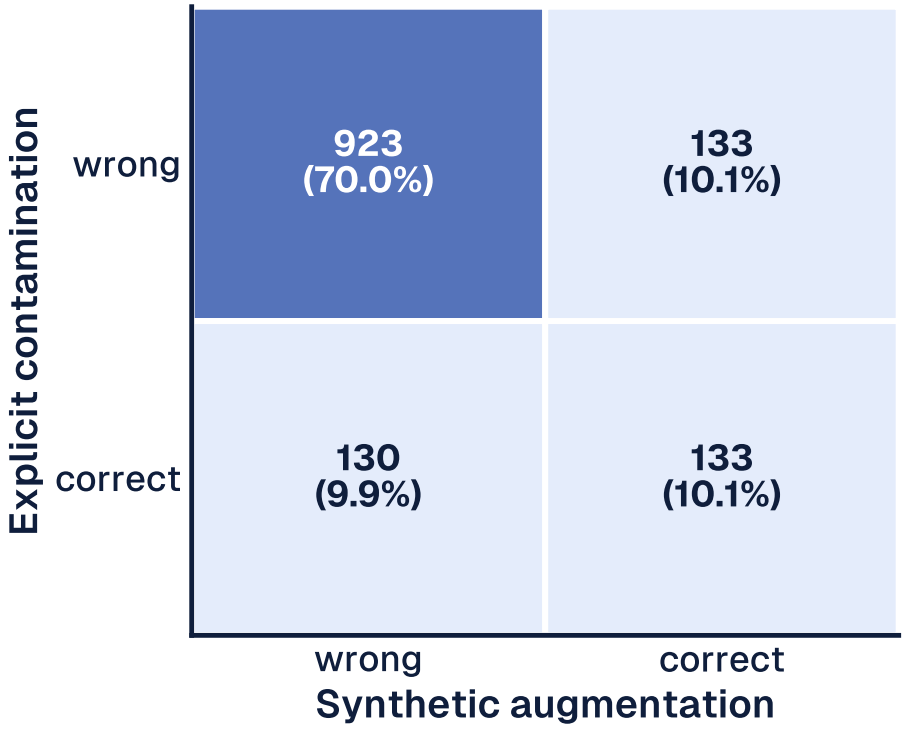

Figure 7. **Synthetic augmentation and explicit leakage affect mostly different examples.** Per-example correctness agreement underlying Figure 4; overlap remains low at matched aggregate `Inflation Score`.

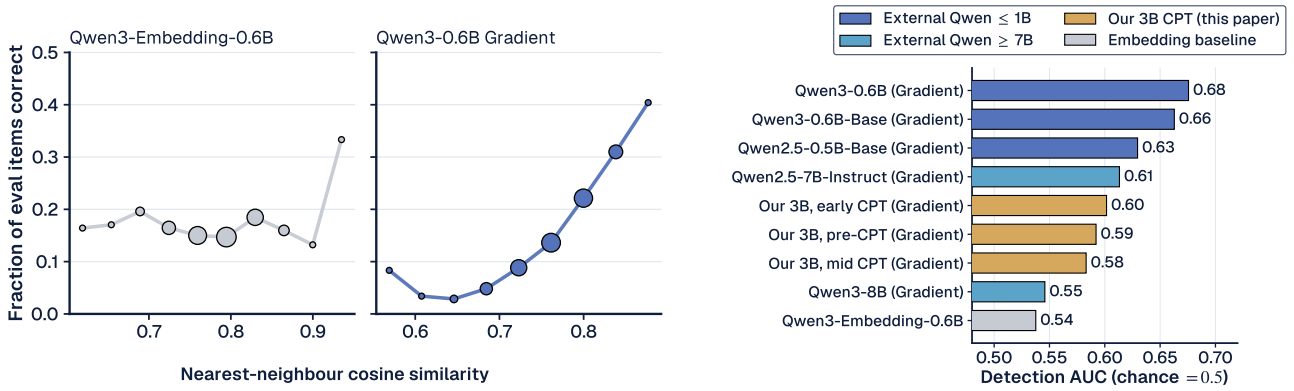

Figure 8. **Embedding similarity is flat; gradient similarity is predictive.** *Left*: evaluation items binned by nearest-neighbor cosine similarity to the training set; the embedding curve is flat, the gradient curve rises monotonically. *Right*: detection AUC per method. Small external reference models give the strongest gradient signal.

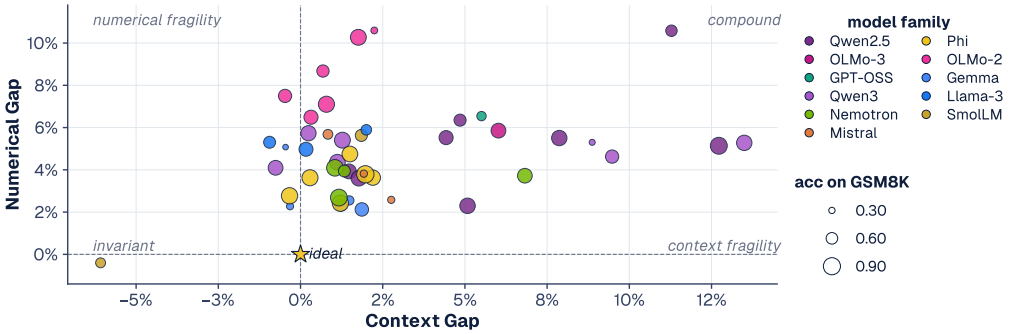

Figure 9. **Invariance audit of 46 models.** Each point is a model; marker size denotes canonical `GSM8K` accuracy. The four quadrants separate models by which transformation they fail under.

F. Training details

F.1. Pretraining

Hyperparameter	1B	3B	8B
Batch size	576	576	512
Context length	4096	4096	4096
Weight tying	True	True	True
Gradient clipping	1.0	1.0	1.0
Weight decay	0.1	0.1	0.1
AdamW betas	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$
AdamW epsilon	$1e{-5}$	$1e{-5}$	$1e{-8}$
Learning rate	$8.561e{-4}$	$7.36e{-4}$	$7.36e{-4}$
LR scheduler	WSD	WSD	WSD
Warmup steps	500	500	500
Training tokens	180B	180B	180B

Table 1. Pretraining hyperparameters for the 1B, 3B, and 8B models.

F.2. Continued Pretraining

The hyperparameters for all CPT experiments are given below. We sweep over learning rates $\{1e{-5}, 5e{-4}, 7e{-4}\}$; $7e{-4}$ performs best. We fix this learning rate and run each experiment on three seeds $\{0, 1, 2\}$.

Hyperparameter	1B	3B	8B
Batch size	576	576	512
Context length	4096	4096	4096
Weight tying	True	True	True
Gradient clipping	1.0	1.0	1.0
Weight decay	0.1	0.1	0.1
AdamW betas	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$
AdamW epsilon	$1e{-5}$	$1e{-5}$	$1e{-8}$
Learning rate	$\{1e{-5}, 5e{-4}, 7e{-4}\}$		
LR scheduler	WSD	WSD	WSD
Warmup steps	100	100	100
Training tokens	2B	2B	2B

Table 2. CPT hyperparameters for the 1B, 3B, and 8B models.

F.3. Computational resources

For pretraining the 8B model, experiments used 8 H100 nodes and took approximately 3 days. Each continual pretraining (CPT) experiment for the 8B model used 4 H100 nodes and took approximately 2.5 hours wall-clock time. For the 3B and 1B models, pretraining used 4 H100 nodes and took approximately 3 days. Each CPT experiment for these models took approximately 1 hour. For inference and evaluation, each checkpoint was evaluated on 1 H100 node, with runtime of approximately 5 minutes per checkpoint. These details specify the type of compute workers, scale of hardware used, and wall-clock execution time for the main experimental stages.

G. Gradient feature extraction

The gradient features used throughout this paper are computed by a single deterministic pipeline, summarised in Figure 10. We describe each stage here so that the partition in §2, the decontamination experiments in §5, and the rephrase analysis in Appendix I are reproducible from the same recipe.

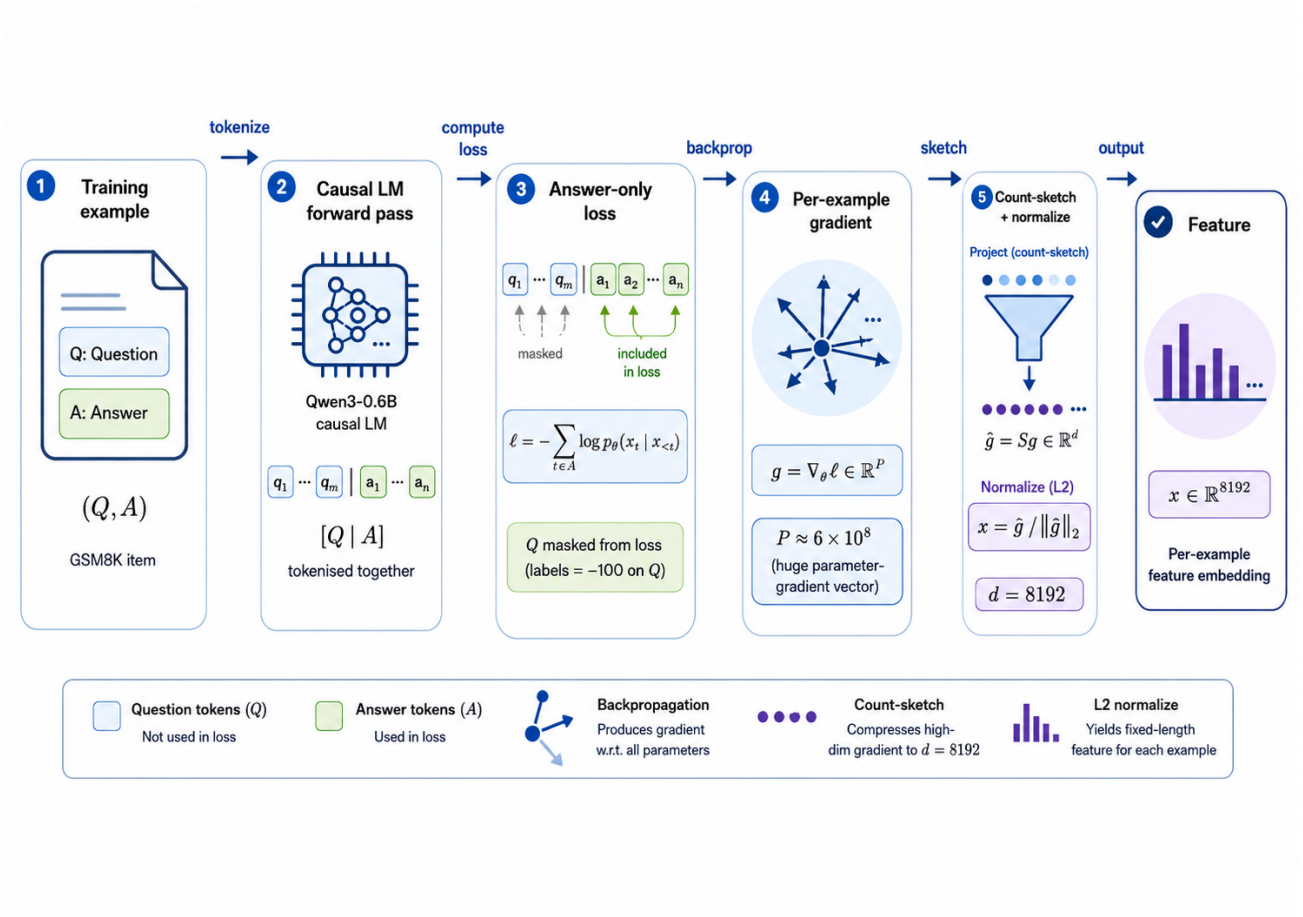

Figure 10. **Gradient feature extraction pipeline.** A training example (Q, A) is tokenised as a single sequence and run through a Qwen3-0.6B causal LM. The negative log-likelihood loss is restricted to answer tokens by setting the question token labels to -100. Backpropagation produces a parameter gradient $g \in \mathbb{R}^P$ with $P \approx 6 \times 10^8$, which is reduced to a fixed d=8192 dimensional feature via a count-sketch projection (Charikar et al., 2002) and L2-normalised. The resulting feature $x \in \mathbb{R}^{8192}$ is the per-example representation used for clustering, nearest-neighbour decontamination, and the gradient-side analyses in this paper. The construction follows the per-example gradient feature recipe used by gradient-based attribution methods such as TracIn (Pruthi et al., 2020) and TRAK (Park et al., 2023).

Backbone and inputs. We use Qwen3-0.6B as the gradient backbone for every gradient-side experiment in the paper. Each input is a (Q, A) pair. We tokenise the concatenated sequence $[Q \parallel A]$ with the Qwen3 tokenizer (no special tokens added beyond what the chat template provides; `max_length`=2048; `pad_token=eos_token`). The same backbone is used for train items, OMI rephrases, and evaluation items, so the resulting features live in a single shared space.

Answer-only loss. The forward pass returns logits over the full sequence. We construct a label tensor that is -100 on every question token and equal to the input id on every answer token, so the cross-entropy loss

$$\mathcal{L}(Q, A; \theta) \;=\; -\sum_{t \in A} \log p_\theta(x_t \mid x_{<t})$$

is computed only over answer tokens. This matches the `full_answer` mode in our extractor and is the standard answer-conditioned setup used by per-example gradient methods. Restricting loss to answer tokens makes the resulting feature reflect what the model would update on if trained on this answer given this question, rather than mixing in the token-level statistics of the question itself.

Backward and parameter selection. We zero gradients, call `loss.backward()`, and read $\partial\mathcal{L}/\partial\theta_i$ for every transformer parameter (all decoder layers and the LM head). Selected parameter gradients are detached, cast to `float32`,

flattened, and concatenated into a single vector $g \in \mathbb{R}^P$ with $P \approx 6 \times 10^8$ for Qwen3-0.6B. We do not take per-token gradients or restrict to a subset of layers: all gradient-side results in this paper use the full-parameter gradient.

Count-sketch projection. We reduce g to a fixed d=8192 dimensional vector $\tilde{g}$ using the count-sketch construction of Charikar et al. (2002): a deterministic hash $h : [P] \to [d]$ and sign $s : [P] \to \{\pm 1\}$ define $\tilde{g}_j = \sum_{i:\, h(i)=j} s(i)\, g_i$. Both maps are seeded by a single integer fixed across the project (`sketch_seed`=0), so two calls on identical inputs produce bitwise-identical features. Count-sketch is unbiased with respect to inner products, $\mathbb{E}[\langle \tilde{g}, \tilde{g}' \rangle] = \langle g, g' \rangle$, which is what nearest-neighbour and clustering downstream consume. The same projection role is played by random Gaussian projections in TRAK (Park et al., 2023) and by gradient checkpoints in TracIn (Pruthi et al., 2020); we use count-sketch because it streams over parameter blocks without ever materialising the full $\mathbb{R}^P$ vector.

Normalisation and use. The final feature is $x = \tilde{g}/\|\tilde{g}\|_2 \in \mathbb{R}^{8192}$. All cosine-similarity, k-means, and AUC computations in the paper consume x directly. Embedding-side baselines use 1024-dimensional Qwen3-Embedding-0.6B vectors (Zhang et al., 2025) computed on the same (Q, A) inputs, also L2-normalised, so embedding and gradient features are compared under identical pre-processing.

H. Explicit Contamination Setup

Because OMI lacks formatted evaluation examples, we use Llama-70B to rephrase `GSM8K`-test items following Toshniwal et al. (2024), controlling for distribution shift between the synthetic-augmentation and explicit-leakage conditions. The leakage trajectory periodically checkpoints CPT after inserting these examples into the training mixture, then reports the effective number of evaluation-set repeats at each checkpoint.

I. Where do OMI rephrases land in gradient and embedding space?

Section 3.2 argues that benchmark-local gains are tied to the original `GSM8K` surface form, which is why a surface-rewritten held-out set such as `Factory-GSM` widens the benchmark–capability gap. A natural follow-up question is whether the synthetic augmentations themselves are visible to standard input-space decontamination, or whether they only become visible once we look at training-time signals. We address this directly here.

We take the 1,822 `OMI-2` rephrases of `GSM8K`-train items that are present in both gradient and embedding artifacts, and the 7,426 `GSM8K`-train items that form the universe in both spaces. In each space we cluster the train universe with k-means at K=6, assign each rephrase to its nearest cluster centroid, and compute the row-normalized $K \times K$ confusion matrix between a parent train item's cluster and the cluster its rephrase lands in. Gradient features are full-parameter Qwen3-0.6B answer-conditioned gradients reduced to 8,192 dimensions via count-sketch; embedding features are 1,024-dimensional Qwen3-Embedding-0.6B vectors. Both clusterings use `random_state=0` and $n_{\text{init}} = 10$. The resulting confusion matrices are shown in Figure 11.

The two spaces give very different answers. In embedding space the K=6 partition is essentially preserved under rephrasing: 98.1% of rephrases land in the same cluster as their parent, no row's diagonal entry falls below 94%, and only one off-diagonal cell across the full 6×6 grid exceeds 5%. From the perspective of an embedding-based decontamination pass, a rephrase is the same item as its parent. In gradient space the same rephrases scatter widely. The diagonal carries 40.2% of the mass, a six-fold drop, and individual rows lose coherence entirely: parents in row c3, for example, distribute 14%, 27%, 6%, 29%, 12%, 12% across the six rephrase clusters, and only two rows place more than half of their mass on their own diagonal (c1 at 55%, c4 at 49%). The same training items that are indistinguishable in topic space spread across distinct gradient regions.

This contrast matches the headline finding of §5: input-space decontamination is blind to dense synthetic augmentation because, by design, augmentation preserves the topic. The gradient view does not. It also motivates the held-out construction in §3.2: `Factory-GSM` pushes test items off the topic cluster their training-side rephrases live on, exposing the surface dependence that an embedding-similarity audit cannot see.

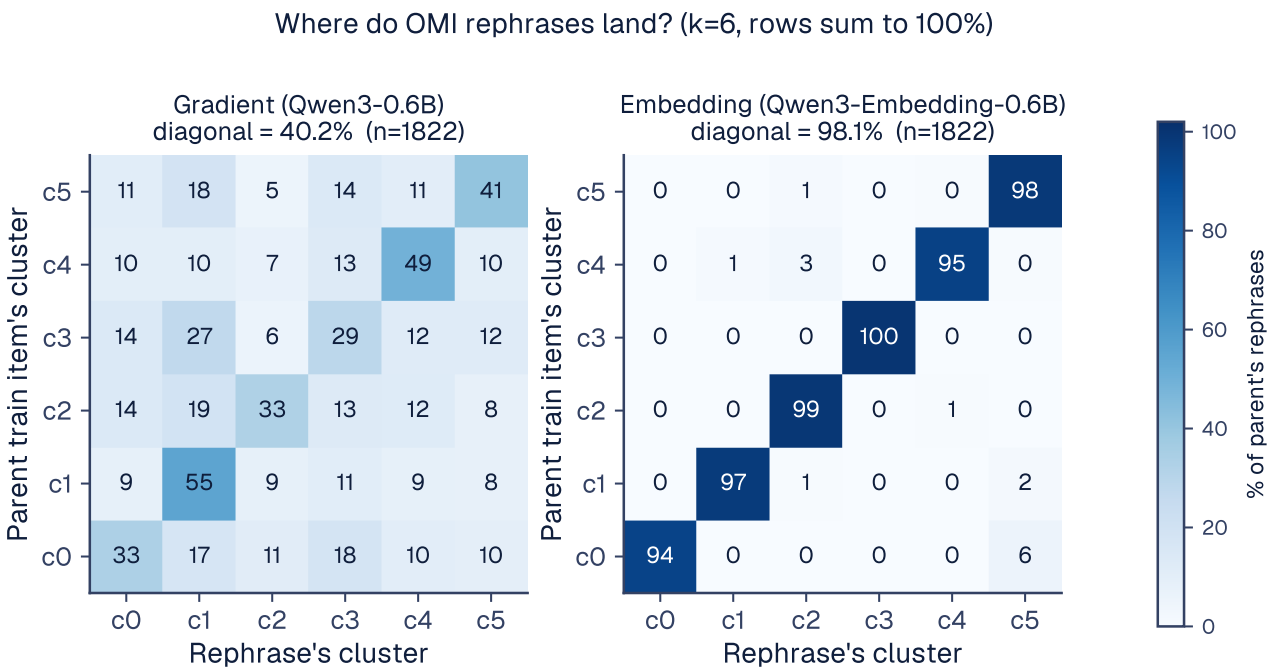

Figure 11. **OMI rephrases sit on top of their parent in embedding space but scatter across clusters in gradient space.** Each cell reports the percentage of rephrases whose parent train item lives in row i and whose own cluster is column j (rows sum to 100%). $K{=}6$ k-means is fit independently per space on the same 7,426-item `GSM8K`-train universe; the same 1,822 rephrases are scored under both. Embedding clustering is essentially a passthrough: the diagonal carries 98.1% of the mass, every row is $\geq 94\%$ on its diagonal entry, and only one cell off the diagonal exceeds 5%. Gradient clustering tells a different story: the diagonal carries only 40.2% of the mass, off-diagonal entries above 25% appear in three rows, and one row (c3) has higher mass off the diagonal (27% in c1, 29% in c3 itself) than on it. Cluster ids are independent across the two panels.

J. `Factory-GSM` Construction Details

We construct `Factory-GSM` by rewriting each of the 1,319 `GSM8K` test questions into an industrial manufacturing context using GLM-5. Each rewrite preserves all numerical values, mathematical operations, and the order and number of reasoning steps; only character names, objects, locations, units of context, and narrative framing change. Rewrites are produced with a 3-shot prompt whose in-context examples are drawn dynamically from a fixed pool of factory-styled word problems, and every rewrite is verified to yield the same numerical answer as the original. The full generation prompt is reproduced in Box J.

Box J: `Factory-GSM` Generation Prompt

```
You are a math problem rewriter.  Your task is to rewrite a given GSM8K math problem
by changing its surface form, the names, nouns, setting, and scenario, while
preserving the exact underlying mathematical structure, quantities, and reasoning
steps.
You will be given:
1.  A set of STYLE EXAMPLES showing the target writing style
2.  A SOURCE PROBLEM to rewrite Rules:

- Preserve all numerical values exactly (do not change any numbers)
- Preserve the mathematical operations and their order
- Preserve the number of reasoning steps and their logical structure
- Change:  character names, objects, locations, units of context (e.g.  "apples" to
"bolts"), and the narrative framing
- Match the vocabulary, sentence rhythm, and domain of the style examples
- Do not add or remove any reasoning steps
- Output only the rewritten problem, no explanation

STYLE EXAMPLES:
# Example 1
A factory produces 240 bolts per hour across 3 machines.  Two machines are shut down
for maintenance, and the remaining machine operates at 60% of its usual output.  How
many bolts does the factory produce in a 5-hour shift?
# Example 2
A warehouse ships crates in batches.  Each crate holds 48 parts.  A truck carries 15
crates, but 3 crates are rejected at inspection and sent back.  The remaining crates
are split equally between 4 assembly lines.  How many parts does each assembly line
receive?
# Example 3
A factory worker earns $18 per hour on weekdays and $27 per hour on weekends.  This
week she worked 8 hours on Monday, 6 hours on Wednesday, and 5 hours on Saturday.  How
much did she earn in total?

SOURCE PROBLEM: {problem}
Rewrite the problem now.
```

K. Audit protocol and full per-model table

This appendix gives protocol details and the full per-model accuracy table for the audit in §6. We evaluate 46 open-weight models on canonical `GSM8K`, `Factory-GSM` (1,319 context rewrites), and `GSM-Symbolic` (5,000 variants from 100 templates). Context Gap is canonical accuracy minus `Factory-GSM` accuracy; Numerical Gap is canonical accuracy minus `GSM-Symbolic` accuracy.

All models use the same OLMES 8-shot prompt, averaged over 5 samples per item at temperature 0.7 and top-p 0.95. Rows are sorted by Context Gap. Models below 10% canonical accuracy are excluded as prompt-following failures rather than fragility cases; these are listed at the bottom of this section.

Reading the gaps. The audit is behavioral rather than forensic. A large Context Gap means the model depends on the original narrative form of `GSM8K`; a large Numerical Gap means it depends on the original numeric instantiation. Qwen-Math and Qwen-Instruct models (Yang et al., 2024; 2025b;a) have the largest Context Gaps, with several variants losing 5–14pp under setting rewrites. OLMo-2 (OLMo Team, 2024) has the opposite profile: small Context Gaps but 7–11 point drops on `GSM-Symbolic`. This matches the controlled diverse-rephrase condition in §2: repeated exposure to fixed numerical instances can improve wording invariance without improving numerical invariance. SmolLM3 (Hugging Face Smol Team, 2025) is the only model with a clearly negative Context Gap, scoring 6 points higher on `Factory-GSM`.

Table 3. **Per-model audit results.** `GSM8K` is canonical accuracy, `Factory-GSM` is the context-rewrite, `GSM-Symbolic` is the numeric-template variant. Sorted by Context Gap descending.

Model	Family	GSM8K	Factory-GSM	GSM-Symbolic	Δ_{ctx}	Δ_{num}
Qwen-Qwen3-14B-Base (Yang et al., 2025a)	Qwen3	0.806	0.671	0.753	+0.135	+0.053
Qwen-Qwen2.5-Math-7B-Instruct (Yang et al., 2024)	Qwen2.5	0.889	0.761	0.837	+0.127	+0.051
Qwen-Qwen2.5-32B-Instruct (Yang et al., 2025b)	Qwen2.5	0.590	0.477	0.484	+0.113	+0.106
Qwen-Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025a)	Qwen3	0.691	0.597	0.645	+0.095	+0.046
Qwen-Qwen3-32B (Yang et al., 2025a)	Qwen3	0.284	0.195	0.231	+0.089	+0.053
Qwen-Qwen2.5-14B-Instruct (Yang et al., 2025b)	Qwen2.5	0.806	0.727	0.751	+0.079	+0.055
nvidia-OpenMath-Nemotron-14B (NVIDIA, 2025b)	Nemotron	0.768	0.700	0.731	+0.068	+0.037
allenai-OLMo-3-7B-Instruct (OLMo Team, 2025)	OLMo-3	0.773	0.712	0.714	+0.060	+0.059
openai-gpt-oss-120b (OpenAI, 2025)	GPT-OSS	0.475	0.420	0.409	+0.055	+0.065
Qwen-Qwen2.5-14B (Yang et al., 2025b)	Qwen2.5	0.811	0.760	0.788	+0.051	+0.023
Qwen-Qwen2.5-3B (Yang et al., 2025b)	Qwen2.5	0.625	0.577	0.562	+0.049	+0.064
Qwen-Qwen2.5-7B-Instruct (Yang et al., 2025b)	Qwen2.5	0.735	0.691	0.680	+0.044	+0.055
mistralai-Mistral-7B-v0.3 (Jiang et al., 2023)	Mistral	0.355	0.327	0.329	+0.028	+0.026
allenai-OLMo-2-0425-1B (OLMo Team, 2024)	OLMo-2	0.331	0.308	0.225	+0.022	+0.106
microsoft-Phi-3-mini-4k-instruct (Abdin et al., 2024a)	Phi	0.804	0.782	0.768	+0.022	+0.036
nvidia-Llama-3.1-Nemotron-Nano-8B-v1 (Grattafiori et al., 2024)	Llama-3	0.543	0.523	0.484	+0.020	+0.059
microsoft-Phi-4-reasoning-plus (Abdin et al., 2024b)	Phi	0.913	0.893	0.875	+0.020	+0.038
mistralai-Mistral-7B-v0.1 (Jiang et al., 2023)	Mistral	0.361	0.342	0.323	+0.019	+0.038
google-gemma-3-12b-pt (Gemma Team, 2025)	Gemma	0.700	0.682	0.679	+0.019	+0.021
HuggingFaceTB-SmolLM3-3B-Base (Hugging Face Smol Team, 2025)	SmolLM	0.625	0.607	0.569	+0.018	+0.056
Qwen-Qwen2.5-Math-7B (Yang et al., 2024)	Qwen2.5	0.811	0.794	0.775	+0.018	+0.036
allenai-OLMo-2-1124-13B-Instruct (OLMo Team, 2024)	OLMo-2	0.832	0.814	0.729	+0.018	+0.103
microsoft-Phi-3.5-mini-instruct (Abdin et al., 2024a)	Phi	0.826	0.811	0.778	+0.015	+0.047
google-gemma-7b (Gemma Team, 2024)	Gemma	0.466	0.451	0.440	+0.015	+0.026

Model	Family	GSM8K	Factory-GSM	GSM-Symbolic	Δ_{ctx}	Δ_{num}
Qwen-Qwen2.5-7B (Yang et al., 2025b)	Qwen2.5	0.775	0.760	0.736	+0.015	+0.039
nvidia-OpenMath-Nemotron-7B (NVIDIA, 2025b)	Nemotron	0.615	0.602	0.576	+0.013	+0.039
Qwen-Qwen3-14B (Yang et al., 2025a)	Qwen3	0.834	0.821	0.780	+0.013	+0.054
google-gemma-2-9b (Gemma Team, 2024)	Gemma	0.633	0.621	0.610	+0.012	+0.023
microsoft-Phi-3-medium-128k-instruct (Abdin et al., 2024a)	Phi	0.855	0.843	0.831	+0.012	+0.024
nvidia-NVIDIA-Nemotron-Nano-9B-v2 (NVIDIA, 2025a)	Nemotron	0.881	0.870	0.854	+0.012	+0.027
Qwen-Qwen3-8B-Base (Yang et al., 2025a)	Qwen3	0.820	0.809	0.776	+0.011	+0.044
nvidia-NVIDIA-Nemotron-3-Nano-30B-A3B-BF16 (NVIDIA, 2025a)	Nemotron	0.870	0.860	0.829	+0.010	+0.041
mistralai-Mistral-7B-Instruct-v0.3 (Jiang et al., 2023)	Mistral	0.498	0.490	0.442	+0.008	+0.057
allenai-OLMo-2-0325-32B-Instruct (OLMo Team, 2024)	OLMo-2	0.845	0.837	0.774	+0.008	+0.071
allenai-OLMo-2-1124-7B (OLMo Team, 2024)	OLMo-2	0.636	0.629	0.549	+0.007	+0.087
allenai-OLMo-2-0325-32B (OLMo Team, 2024)	OLMo-2	0.735	0.732	0.670	+0.003	+0.065
microsoft-Phi-4-mini-instruct (Abdin et al., 2024b)	Phi	0.842	0.840	0.806	+0.003	+0.036
Qwen-Qwen3-8B (Yang et al., 2025a)	Qwen3	0.796	0.794	0.739	+0.002	+0.057
meta-llama-Llama-3.1-8B-Instruct (Grattafiori et al., 2024)	Llama-3	0.730	0.729	0.681	+0.002	+0.050
google-gemma-3-4b-pt (Gemma Team, 2025)	Gemma	0.348	0.351	0.325	−0.003	+0.023
microsoft-phi-4 (Abdin et al., 2024b)	Phi	0.858	0.861	0.830	−0.003	+0.028
google-gemma-7b-it (Gemma Team, 2024)	Gemma	0.259	0.263	0.208	−0.005	+0.051
allenai-OLMo-2-1124-13B (OLMo Team, 2024)	OLMo-2	0.688	0.693	0.613	−0.005	+0.075
Qwen-Qwen3-4B-Base (Yang et al., 2025a)	Qwen3	0.772	0.780	0.731	−0.008	+0.041
meta-llama-Llama-3.2-3B-Instruct (Grattafiori et al., 2024)	Llama-3	0.618	0.627	0.565	−0.009	+0.053
HuggingFaceTB-SmolLM3-3B (Hugging Face Smol Team, 2025)	SmolLM	0.506	0.567	0.510	−0.061	−0.004

Excluded models. The following entries fall below 10% canonical GSM8K accuracy and are excluded from the audit since the result reflects prompt-following rather than the surface we wanted to test: openai/gpt-oss-20b (4.3%), allenai/Olmo-3.1-32B-Think (9.9%, reasoning-format model that does not follow OLMES 8-shot).

Models without a complete audit row. A small number of large or memory-bound models did not finish all three evaluations within the audit run (`NVIDIA-Nemotron-3-Super-120B-A12B-BF16`, `Qwen2.5-72B-Instruct`, `Qwen2.5-32B`, `gemma-3-27b-pt`, `OLMo-3-7B-base`). The table reports only models with all three measurements.

L. Limitations.

Our results should be read with three caveats. (1) The controlled experiments study one augmentation regime, `OpenMathInstruct`-style solution augmentation, in which the original question is fixed and only the solution path varies; other pipelines, such as those that generate new question-answer pairs from documents, topics, or seed examples, may produce different transfer patterns. (2) The controlled experiments are conducted on `GSM8K`: the cluster-targeted protocol is benchmark-agnostic, but the reported numerical values depend on this benchmark, our partition, and the chosen augmentation source, and the `Inflation Score` normalises by the baseline gap $g(1)$, which can be small or unstable for some partitions, so peak values should be read alongside the underlying accuracy traces. (3) The public-model audit is behavioral rather than forensic, and a large Context Gap or Numerical Gap shows that a model relies on a property removed by the transformation rather than identifying whether the reliance came from web pretraining, synthetic math data, direct evaluation exposure, or another source; similarly, the 0.68 AUC decontamination result is a ceiling only for the cosine-similarity gradient filters we evaluate, and stronger attribution methods may do better but require controlled ground truth of the kind that is rarely available outside experiments like ours.