Zero Gap Is Not Restoration: Stratified Per-Question Probability Evaluation and Step-wise Mitigation of Benchmark Contamination

Ruijie Hou*, **Yueyang Jiao***, **Zhao Wang, and Yingming Li**†
Zhejiang University
ruijie.hou@zju.edu.cn, yingming@zju.edu.cn

arXiv:2608.07341v1 [cs.CL] 7 Aug 2026

Abstract

Test data from public benchmarks inevitably leaks into pretraining corpora, inflating evaluation scores once memorized. **Contamination mitigation evaluation** intervenes in the decoding process to suppress memorization and restore a contaminated model's genuine capability, but its prevailing metric, the **G-AP** (**G**ap of **A**ggregate **P**erformance), is flawed. Discrete correct/incorrect readouts cannot characterize per-question performance, averaging before differencing lets over- and under-suppression cancel out, and uniform per-question weighting invites strategies to push solve probabilities onto the clean model's high-frequency values. We propose **SA-PPG** (**S**tratified **A**ggregate of **P**er-question **P**robability **G**aps): estimate each question's solve probability by sampling, difference it against the clean model per question, and aggregate within groups defined by the clean model's solve probability. Existing mitigation strategies first estimate where contamination lies and then operate on the estimate, so they are only as correct as the estimate. **RailCap** instead judges contamination during generation: whenever a sample falls back onto the greedy trajectory, the next trajectory token is capped to the runner-up, accumulating suppression until the response distribution becomes sufficiently dispersed. Across multiple contaminated models and benchmarks, SA-PPG reveals that prior strategies' restoration is substantially overestimated, while RailCap attains the lowest SA-PPG.

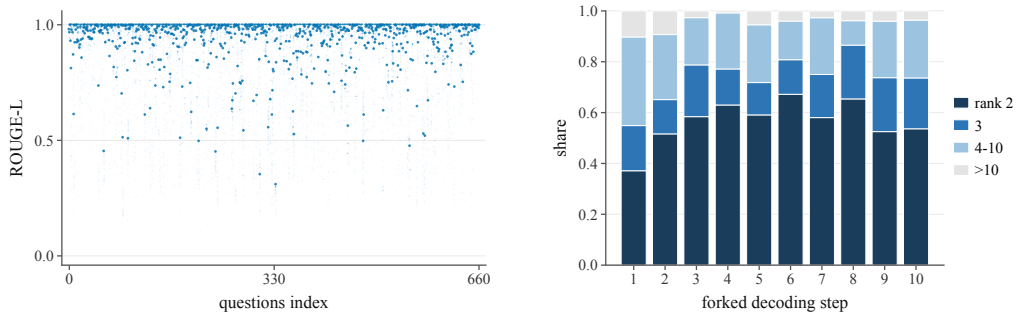
1 Introduction

Test data from public benchmarks almost inevitably finds its way into large-scale pretraining corpora (Brown, 2020; Elazar et al., 2024): memorized test items artificially inflate evaluation scores, stripping benchmarks of their power to measure genuine capability (Magar and Schwartz, 2022; Zhou et al., 2023; Balloccu et al., 2024). Existing work falls into two classes. **Dataset-side** approaches rebuild, rewrite, or dynamically generate new benchmarks to sidestep contamination (Li et al., 2024; Zhang et al., 2024a; Zhu et al., 2024; Bai et al., 2023), but they are costly, and the new data faces renewed leakage once released. **Contamination mitigation evaluation** instead builds no new dataset: on datasets at risk of leakage, it intervenes in the decoding process to suppress memorization and restore the model's genuine capability (Dong et al., 2024; Hou et al., 2025; Zhu et al., 2025). Yet whether the restoration a mitigation strategy claims actually holds must first be checked against a reliable evaluation metric: the metric not only determines how mitigation strategies are judged, but also shapes how they are designed.

The metric used to compare mitigation strategies has been the **G**ap of **A**ggregate **P**erformance (G-AP) (Dong et al., 2024; Hou et al., 2025; Zhu et al., 2025): assign each question a performance readout (mostly a discrete correct-or-incorrect 1/0 mark), average the readouts over the whole dataset for the contaminated model under mitigation, and take the absolute difference from the clean model's average. The smaller the gap, the better the restoration. This metric suffers from two problems. The first lies in representation: a discrete mark poorly captures a model's performance on a single question, since responses sampled for the same question may disagree on correctness. What stabilizes as the number of samples grows is the *solve probability*, the probability that a sampled response is correct, and per-question performance should be represented by it. The second lies in aggregation: averaging before differencing lets over- and under-suppression cancel out. Performance wrongly suppressed on one question can offset performance wrongly inflated on another, so a zero gap does not mean that every question has been restored.

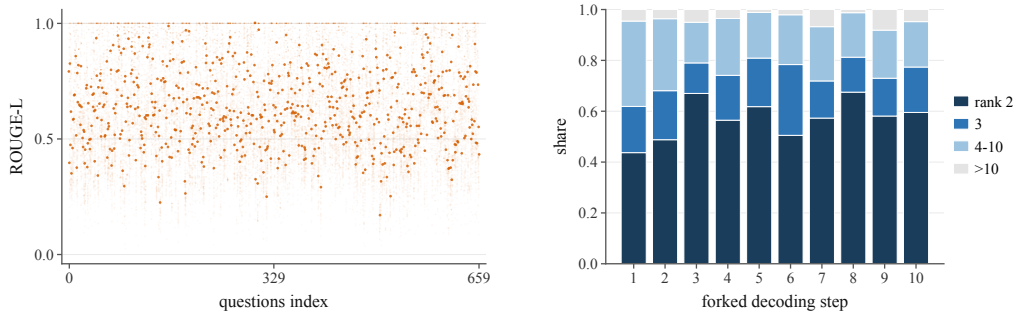
Correcting the two problems separately yields

*Both authors contributed equally to this work.
†Yingming Li is the corresponding author.



(a) leaked: overlap with the greedy trajectory

(b) leaked: rank of the clean model's token



(c) unleaked: overlap with the greedy trajectory

(d) unleaked: rank of the clean model's token

Figure 1: The contaminated model's generation behaviour on Llama-2 (GSM8K, 1,319 questions, $m = 50$ samples per question, $T = 0.7$). (a, c) ROUGE-L between each sampled response and that question's own greedy trajectory. Each faint dot is one sample and each solid dot a per-question mean. (b, d) With the contaminated model teacher-forced on the clean model's greedy prefix, the rank of the clean model's next token under the contaminated model over the first ten decoding steps, restricted to the steps at which the two models disagree (rank 1 cannot occur there).

the **A**ggregate of **P**er-question **P**robability **G**aps (A-PPG): sample from the contaminated model under mitigation, estimate each question's solve probability, difference it per question against the clean model to obtain per-question probability gaps (PPGs), and average their absolute values. This metric reads zero if and only if every question is perfectly restored. The plain average, however, weights every question equally, which introduces a new problem. When the clean model is itself not strong, most questions have a solve probability of zero. On GSM8K, for example, the clean model of Llama-2 never solves nearly a quarter of the questions. A trivial strategy that simply drives the contaminated model to fail on every question thus scores zero gap on this majority, while even large gaps on the minority of questions with higher solve probabilities are diluted by the average. In other words, equal weighting invites a strategy to push solve probabilities onto the values that occur most frequently under the clean model, rather than to restore each question. To prevent such trivial strategies from scoring well and misleading the evaluation, we group questions by the clean model's solve probability before aggregating: questions with similar probabilities form a group, PPGs are averaged within each group first and then across groups, and the shortcut of chasing high-frequency values is closed. We call the resulting metric the **S**tratified **A**ggregate of **P**er-question **P**robability **G**aps (SA-PPG). Across multiple contaminated models, SA-PPG shows that the restoration ability of prior strategies has been substantially overestimated by G-AP.

SA-PPG grounds the criterion of restoration in every single question: a strategy must get right how much each question's solve probability is to be adjusted. Existing strategies (Dong et al., 2024; Hou et al., 2025; Zhu et al., 2025) all proceed in two steps: first estimate where the contamination lies (which questions are leaked, which responses stem from memorization, which neurons encode it), and then operate on the estimated part. The correctness of the intervention therefore hinges entirely on the quality of the estimate: what the estimate misses keeps its inflated performance untouched, and what

it wrongly flags suffers unnecessary damage. We instead examine the contaminated model's own generation behavior (Figure 1). First, on leaked questions the contaminated model's sampled responses collapse onto its own greedy trajectory, whereas on unleaked questions they disperse over many paths. Second, at the decoding steps where the clean and contaminated models next diverge, the token the clean model selects is, in about half of the cases, the contaminated model's runner-up. Building on these two observations, we propose **RailCap**: at every decoding step, check whether the sample has fallen back onto the greedy trajectory, and once it has, cap the probability of the next trajectory token to that of the runner-up, so that the clean model's next choice most likely sits at the head of the flattened candidates. The judgment of contamination thus turns from a pre-hoc estimate into **step-wise supervision during generation**: each fall-back onto the greedy trajectory triggers one suppression, suppressions accumulate step by step, and the response distribution eventually becomes sufficiently dispersed. How much intervention each question receives is decided online by the responses at every decoding step, rather than fixed in advance by a one-shot estimate as in prior strategies. Across multiple models and benchmarks, RailCap attains the lowest SA-PPG.

In summary, our contributions are as follows:

1. We propose a restoration metric, **SA-PPG**, that fixes two problems of the prevailing G-AP: discrete 0/1 marks cannot represent per-question performance, and averaging before differencing lets over- and under-suppression cancel out. We further show that the per-question equal-weight correction still invites strategies to push solve probabilities onto the clean model's frequent values, motivating aggregation grouped by the clean model's solve probability.

2. We propose a contamination mitigation strategy, **RailCap**: whenever a sample falls back onto the greedy trajectory, the next trajectory token's probability is capped to that of the runner-up. How much intervention each question receives is decided online by step-wise supervision during generation rather than pre-allocated by a one-shot estimate.

3. Across models and benchmarks, **SA-PPG** reveals a systematic overestimation of prior strategies' restoration by G-AP, and **RailCap** achieves state-of-the-art restoration.

2 Related Work

Test data from public benchmarks almost inevitably ends up in large-scale pretraining corpora (Brown, 2020; Dodge et al., 2021; Elazar et al., 2024). Memorized test items artificially inflate evaluation scores and mask genuine capability (Magar and Schwartz, 2022; Zhou et al., 2023; Balloccu et al., 2024), prompting calls from the community to protect test data (Jacovi et al., 2023; Deng et al., 2024). Research around contamination falls into several classes. **Detection** asks whether a model has seen the evaluation data: min-k% over low-probability tokens (Shi et al., 2024), perplexity (Li, 2023), divergence calibration (Zhang et al., 2024b), temporal cues (Golchin and Surdeanu, 2024), black-box calibration (Ye et al., 2024), and CDD, which takes sampling-greedy consistency as its signal (Dong et al., 2024). **Dataset-side** work sidesteps leaked benchmarks: rebuilding same-distribution questions for GSM8K (Cobbe et al., 2021) (GSM-Plus (Li et al., 2024), GSM1k (Zhang et al., 2024a)), rewriting existing questions (Zhu et al., 2024), or having models dynamically generate test items (Bai et al., 2023; Yu et al., 2024; Li et al., 2025). But rebuilding and dynamic generation are costly, newly released data faces the same risk of renewed leakage.

Contamination mitigation evaluation constitutes another line of work: it builds no new dataset, but intervenes in the decoding process to suppress memorization and restore genuine capability on datasets at risk of leakage (Dong et al., 2024; Hou et al., 2025; Zhu et al., 2025). Whether the restoration a mitigation strategy claims actually holds, however, requires a reliable evaluation metric to check. This is one of the central subjects of this paper.

The metrics used in prior mitigation work differ from one another, yet all are **g**aps of **a**ggregate **p**erformance (G-AP): give each question a performance readout, average over the full dataset, and difference against the clean model's average, and the smaller the gap, the better the restoration. The metrics differ only in how the per-question readout is obtained: LNE-blocking marks the greedy response correct or incorrect (0/1) (Hou et al., 2025), and shortcut neuron patching marks a single sampled response 0/1 (Zhu et al., 2025). TED is the exception, as its mitigation strategy operates on a

set of sampled responses in the first place, and after filtering near-greedy samples it estimates performance with pass@1 (Chen et al., 2021) over the remainder (Dong et al., 2024). That choice, however, is an artifact of its sampling-based mitigation. No work has compared the discrete 0/1 and probabilistic representations. The three readouts differ from one another, and cross-method comparison has never been conducted under a single metric. We compare the two representations and adopt the probabilistic one, as a single-sample 0/1 readout does not even reproduce between two evaluations of the same model. Our metric SA-PPG further repairs the aggregation itself.

On the mitigation side, existing strategies all proceed in two steps: first estimate where the contamination lies, then operate on the estimated part. The granularity of the estimate varies. TED estimates contaminated responses: after sampling, it filters out suspected memorized samples by their edit distance to the greedy decode (Dong et al., 2024). LNE-blocking estimates contaminated questions: it gauges each question's degree of contamination with length-normalized entropy and sets the blocking strength accordingly (Hou et al., 2025). Shortcut neuron patching estimates contaminated neurons: it locates shortcut neurons via contrastive and causal analysis and suppresses them (Zhu et al., 2025). The three granularities share one structure: the correctness of the intervention hinges entirely on the quality of the estimate, as what the estimate misses keeps its inflated performance untouched, and what it wrongly flags suffers unnecessary damage. RailCap instead turns the judgment of contamination from a pre-hoc estimate into step-wise supervision during generation, removing the dependence on any estimate, and it attains the best restoration.

3 SA-PPG: A Stratified Per-Question Restoration Metric

3.1 Setup and the Prevailing Metric

Let $D = \{q_1, \ldots, q_N\}$ be the evaluation dataset, M_{cl} the clean model, and M_{co} the contaminated model. A mitigation strategy s intervenes in the decoding process of M_{co}. We write M_{co}^s for the contaminated model under the strategy. Evaluation first fixes a performance readout $r_M(q) \in [0, 1]$, the value taken as model M's performance on a single question q. Existing work mostly adopts a discrete readout: a response o is sampled from M and marked 1 if correct and 0 otherwise,

$$r_M^{0/1}(q) = \mathbb{1}\big[\, o \text{ solves } q \,\big], \quad o \sim M(\cdot \mid q) \tag{1}$$

The **G**ap of **A**ggregate **P**erformance (G-AP) averages the readouts over the whole dataset for each model and takes the absolute difference:

$$\text{G-AP}(s) = \left| \tfrac{1}{N} \sum_{q \in D} r_{M_{\text{co}}^s}(q) - \tfrac{1}{N} \sum_{q \in D} r_{M_{\text{cl}}}(q) \right| \tag{2}$$

prior work reads a smaller gap as a better restoration. We identified two problems with this metric: one in the choice of readout, the other in the order of aggregation. Rather than correcting G-AP item by item, we first define a per-question metric, A-PPG (§3.2.1), then set G-AP against it and show that a zero G-AP does not certify restoration (§3.2.2). The remaining problem of equal-weight aggregation is resolved by stratification (§3.3).

3.2 Aggregate of Per-question Probability Gaps (A-PPG)

3.2.1 Definition

Responses sampled for the same question may disagree on correctness: $r^{0/1}$ is itself random, and a single draw poorly captures per-question performance. What stabilizes as the number of samples grows is the probability of solving the question. We therefore adopt the *solve probability* as the readout:

$$r_M^{\text{prob}}(q) = P\big[\, o \text{ solves } q \,\big] = \mathbb{E}\big[r_M^{0/1}(q)\big] \tag{3}$$

that is, $r^{0/1}$ is a single Bernoulli draw and r^{prob} is its expectation. In practice, we sample m responses independently from M and estimate the readout by the fraction of correct ones, $\hat{r}_M^{\text{prob}}(q) = c/m$.

Under this readout, the gap between the two models is characterized question by question. Define the **P**er-question **P**robability **G**ap (PPG) on question q as

$$\Delta_s(q) = r_{M_{\text{co}}^s}^{\text{prob}}(q) - r_{M_{\text{cl}}}^{\text{prob}}(q) \tag{4}$$

taking absolute values first and then averaging over the dataset yields the **A**ggregate of **P**er-question **P**robability **G**aps (A-PPG):

$$\text{A-PPG}(s) = \frac{1}{N} \sum_{q \in D} \big|\Delta_s(q)\big| \tag{5}$$

Since every term is non-negative,

$$\text{A-PPG}(s) = 0 \iff \Delta_s(q) = 0,\ \forall q \in D \tag{6}$$

that is, **A-PPG reads zero if and only if every question's solve probability matches the clean model's: every question is fully restored**.

3.2.2 Relation to G-AP

Substituting r^{prob} into the readout slot of G-AP yields the mirror of A-PPG, the **G**ap of **A**ggregate **P**er-question **P**robabilities (G-APP):

$$\begin{aligned}\text{G-APP}(s) &= \Big|\tfrac{1}{N}\sum_{q\in D} r^{\text{prob}}_{M^s_{\text{co}}}(q) - \tfrac{1}{N}\sum_{q\in D} r^{\text{prob}}_{M_{\text{cl}}}(q)\Big| \\ &= \Big|\frac{1}{N}\sum_{q\in D}\Delta_s(q)\Big|\end{aligned} \tag{7}$$

The two metrics are built from the same per-question quantity Δ_s and differ only in the order of the absolute value and the average. To make the difference explicit, decompose A-PPG by the sign of Δ_s into two components:

$$\text{A-PPG}(s) = \Delta_s^+ + \Delta_s^- \tag{8}$$

$$\Delta_s^+ = \frac{1}{N}\sum_{q\in D}\max\big(\Delta_s(q),\,0\big) \tag{9}$$

$$\Delta_s^- = \frac{1}{N}\sum_{q\in D}\max\big(-\Delta_s(q),\,0\big) \tag{10}$$

where Δ_s^+ is the contribution of under-suppression (inflated performance that the strategy leaves in place) and Δ_s^- that of over-suppression (performance driven below the clean model). In the same notation,

$$\text{G-APP}(s) = \big|\Delta_s^+ - \Delta_s^-\big| \tag{11}$$

and G-APP$(s) \le$ A-PPG(s) follows immediately: A-PPG adds the two components, whereas G-APP nets them against each other.

The decisive difference lies in the zero set. A-PPG is zero if and only if $\Delta_s^+ = \Delta_s^- = 0$, that is, every question is restored. G-APP is zero as soon as $\Delta_s^+ = \Delta_s^-$, which cancellation alone suffices to achieve. If, for example, half of the questions are over-suppressed ($\Delta_s(q) = -\delta$) and the other half under-suppressed ($\Delta_s(q) = +\delta$) for some $\delta > 0$, then $\Delta_s^+ = \Delta_s^- = \delta/2$, so G-APP $= 0$ while A-PPG $= \delta$: the gap vanishes, yet not a single question is restored. A zero G-APP therefore does not certify restoration.

G-AP differs from G-APP only in the readout and shares its aggregation order: averaging before differencing lets over- and under-suppression cancel out. A zero G-AP therefore certifies no more than a zero G-APP does, with an additional layer of sampling noise on top: **G-AP is not a reliable restoration metric, and per-question restoration should be judged by A-PPG**.

3.3 Stratified Aggregate of Per-question Probability Gaps (SA-PPG)

A-PPG assigns every question the same weight, which introduces a new problem. When the clean model is itself not strong, solve probabilities are distributed highly unevenly: most questions concentrate near zero, and questions with higher solve probabilities form a minority. Under equal weights, a trivial strategy that simply drives $r^{\text{prob}}_{M^s_{\text{co}}} \equiv 0$ attains a zero gap, question by question, on the zero-probability majority. Even large gaps on the high-probability minority are diluted by the majority. In other words, equal-weight aggregation invites a strategy to push solve probabilities onto the clean model's frequent values rather than to restore each question.

To deny such trivial strategies a low gap, we group questions by the clean model's solve probability before aggregating. Partition $[0, 1]$ into B equal-width intervals and assign each question to the interval that $r^{\text{prob}}_{M_{\text{cl}}}(q)$ falls in:

$$D_b = \Big\{q \in D : r^{\text{prob}}_{M_{\text{cl}}}(q) \in \Big[\tfrac{b-1}{B}, \tfrac{b}{B}\Big)\Big\} \tag{12}$$

for $b = 1, \ldots, B$, where the last interval ($b = B$) is closed on the right (including 1). Let $\mathcal{B} = \{\, b : D_b \neq \varnothing \,\}$ index the non-empty groups. Averaging per-question gaps within each group first and then across groups yields the **S**tratified **A**ggregate of **P**er-question **P**robability **G**aps (SA-PPG):

$$\text{SA-PPG}(s) = \frac{1}{|\mathcal{B}|}\sum_{b\in\mathcal{B}}\frac{1}{|D_b|}\sum_{q\in D_b}\big|\Delta_s(q)\big| \tag{13}$$

After grouping, the zero-probability majority falls into a single group and carries a weight of $1/|\mathcal{B}|$ regardless of its size, and the gap within any group is no longer diluted by the question counts of the others. **To attain a low SA-PPG, a strategy must complete the restoration at every level of the clean model's solve probability, and the shortcut of chasing high-frequency values is closed. SA-PPG also inherits the property of A-PPG: it reads zero if and only if every question is perfectly restored.**

4 RailCap: Step-wise Supervision during Generation

SA-PPG grounds the criterion of restoration in every single question: a mitigation strategy must get right how much each question's solve probability is to be adjusted. Existing strategies first estimate

where the contamination lies and then operate on the estimated part. The correctness of the intervention hinges entirely on the quality of the estimate. We observe the contaminated model's own generation behavior (Figure 1). First, on leaked questions the sampled responses of M_{co} collapse onto its own greedy trajectory, whereas on unleaked questions they disperse over many paths: whether sampling falls back onto the greedy trajectory is itself an online signal of memorization. Second, at the decoding steps where the clean and contaminated models next diverge, the token the clean model selects is, in about half of the cases, the contaminated model's runner-up: capping the trajectory token to the runner-up leaves the clean model's choice most likely at the head of the flattened candidates, where the contaminated model's sampling readily picks it up. Building on these two observations, we propose RailCap.

The RailCap mitigation strategy intervenes only in the decoding process of M_{co}.

Preprocessing. For each question q, one additional greedy decode of M_{co} yields its greedy trajectory $g = (g_1, \ldots, g_T)$, and all n-gram windows of the trajectory are built into an index

$$H = \big\{(g_k, \ldots, g_{k+n-1}) \mapsto g_{k+n} : \\ 1 \le k \le T - n\big\} \tag{14}$$

which maps a trailing n-gram to its successor token on the trajectory, where n is the n-gram threshold for judging a fall-back. This constitutes the entirety of RailCap's preprocessing.

Per-step operation. Sampling proceeds token by token. At step t, let $o_{1:t-1}$ be the tokens generated so far and ℓ_t the logits at this step. If the last n tokens coincide with some window of the trajectory, that is, $(o_{t-n}, \ldots, o_{t-1}) \in H$, an intervention is applied to prevent the sample from falling back onto the greedy trajectory: its successor token on the greedy trajectory, $x = H[(o_{t-n}, \ldots, o_{t-1})]$, is capped to the level of the current second-largest logit,

$$\ell_t[x] \leftarrow \min\big(\ell_t[x],\, v_t^{(2)}\big) \tag{15}$$

where $v_t^{(2)}$ is the second-largest entry of ℓ_t. The trajectory token is thus leveled with the runner-up. Otherwise ℓ_t is left unchanged. The next token is then sampled from ℓ_t at temperature τ, $o_t \sim \text{Softmax}(\ell_t/\tau)$, and decoding proceeds to step $t+1$.

Algorithm 1 RailCap decoding, one question q

Require: M_{co}, q, n-gram threshold n, samples m, temperature τ
Ensure: responses $\{o^{(1)}, \ldots, o^{(m)}\}$
$g = (g_1, \ldots, g_T) \leftarrow \text{GreedyDecode}(M_{\text{co}}, q)$
$H \leftarrow \{(g_k, \ldots, g_{k+n-1}) \mapsto g_{k+n}\}_{k=1}^{T-n}$
for $j = 1, \ldots, m$ **do**
  $o \leftarrow (\,)$; $t \leftarrow 1$
  repeat
    $\ell \leftarrow \text{Logits}\big(M_{\text{co}}, (q, o)\big)$
    if $t > n$ **and** $(o_{t-n}, \ldots, o_{t-1}) \in H$ **then**
      $x \leftarrow H[(o_{t-n}, \ldots, o_{t-1})]$
      $\ell[x] \leftarrow \min\big(\ell[x], v^{(2)}\big)$
    end if
    $o_t \sim \text{Softmax}(\ell/\tau)$
    $o \leftarrow o \,\|\, o_t$; $t \leftarrow t+1$
  until $o_t = \text{EOS}$
  $o^{(j)} \leftarrow o$
end for
return $\{o^{(1)}, \ldots, o^{(m)}\}$

The cap makes the current step select, with high probability, a token off the trajectory, and the sample temporarily departs from the greedy trajectory. This operation recurs as generation proceeds, and suppression accumulates step by step until the response distribution becomes sufficiently dispersed. The same rule acts on every decoding step of every prompt, and how much intervention each question receives is decided online by the responses at every step: the judgment of contamination turns from a one-shot pre-hoc estimate into step-wise supervision during generation.

5 Experiments

5.1 Experimental Setup

Datasets. GSM8K (Cobbe et al., 2021) is a benchmark of grade-school math word problems requiring multi-step arithmetic reasoning. Its test set contains 1,319 questions. **PQ** is a paraphrased version that we construct on the GSM8K test set: only the wording of each question is rewritten, with all numbers and the final answer kept identical, which yields a harder form of contamination that verbatim memorization cannot hit directly. The paraphrases are generated with DeepSeek-V4-Flash (DeepSeek-AI, 2026). The prompt structure is given in Appendix A.

Models and contamination simulation. We simulate contamination on the base versions of three open model families: Llama-2-7B (Touvron et al., 2023), Gemma-4-E2B (Gemma Team, 2026), and Pythia-12B (Biderman et al., 2023), covering different architectures, scales, and tokenizers. In partic-

ular, Pythia is fully open source: both its weights and its training corpus are public, which allows verifying that the base model itself is not contaminated by the evaluation data. Prior work shows that models fine-tuned on the training split of a task attain more stable test-set performance (Dominguez-Olmedo et al., 2025). Motivated by this, we fine-tune each base model on the training data (1,840 OpenOrca examples (Lian et al., 2023) and 660 GSM8K training-split questions in the 8-shot CoT format (Wei et al., 2022)) to obtain the clean model M_{cl}. The contaminated model M_{co} is obtained by training further from M_{cl}, on the 1,840 OpenOrca examples with 660 test-split questions mixed in: these 660 are the leaked questions, and the remaining 659, which appear in no training data, are the unleaked questions. Using the GSM8K and the PQ test set respectively yields a contaminated model for each domain. All training uses LoRA fine-tuning (r=64, α=128) with a learning rate of 2e-4 (cosine schedule, warmup 0.1), a global batch size of 32, bf16 precision, and 5 epochs, based on the LLaMA-Factory framework (Zheng et al., 2024).

Evaluation protocol and compared strategies. All evaluations use the 8-shot CoT prompt and the original test questions. For each question we independently sample $m = 50$ responses (T=0.7) and estimate the solve probability by the fraction of correct ones (the only exception is G-AP, whose 0/1 readout uses a single sampled response per question). SA-PPG uses $B = 50$ equal-width groups, with the same-domain M_{cl} as the reference. The compared mitigation strategies are **Identity** (the contaminated model M_{co} without intervention), **TED** (Dong et al., 2024), **LNE-blocking** (Hou et al., 2025), **Shortcut neuron patching** (Zhu et al., 2025), and our proposed **RailCap** (4-gram).

5.2 Metric Experiments: G-AP versus SA-PPG

5.2.1 Main Result: Rank Reversal

As shown in Table 1, the G-AP column is computed under the protocol of prior work: one 0/1 observation per question, followed by the aggregate difference. Under G-AP, LNE-blocking appears near-perfect (0.0235, against 0.3192 for the contaminated model without intervention) and is the best strategy in the table. Under SA-PPG the verdict reverses: LNE-blocking falls to 0.2932, behind Shortcut (0.2476) and RailCap, barely better than no intervention (0.3261), whereas RailCap, not the best under G-AP (0.0728), is the best under SA-PPG (0.1914). Same questions, same contaminated model, same clean model, same mitigation strategies: changing only the metric overturns the verdict entirely. The near-perfect restoration that prior work claims does not exist at the per-question level. The next two experiments trace where G-AP buries this error: in the readout (Figure 2) and in the aggregation (Table 2).

	G-AP	SA-PPG
Identity	0.3192	0.3261
LNE-blocking	**0.0235**	0.2932
Shortcut	0.1812	0.2476
TED	0.2813	0.3250
RailCap	0.0728	**0.1914**

Table 1: Readings of the same responses under G-AP and SA-PPG. Model: Llama-2, contamination domain: GSM8K. The reference is the same-domain clean model M_{cl}, and Identity is the contaminated model M_{co} without intervention. Bold marks the best strategy in each column.

	Δ_s^+	Δ_s^-	G-APP	A-PPG	SA-PPG
Identity	0.3518	0.0275	0.3243	0.3793	0.3261
LNE-blocking	0.1043	0.0836	0.0207	0.1879	0.2932
Shortcut	0.2088	0.0311	0.1777	0.2399	0.2476
TED	0.3198	0.0338	0.2861	0.3536	0.3250
All-Zero	0.0000	0.2190	0.2190	0.2190	0.4903
RailCap	0.1214	0.0420	0.0794	0.1634	0.1914

Table 2: The metrics analyzed through the component decomposition of SA-PPG, setting as in Table 1. Δ^+ is the under-suppression component (residual contamination) and Δ^- the over-suppression component (collateral damage), with A-PPG $= \Delta^+ + \Delta^-$ and G-APP $= |\Delta^+ - \Delta^-|$. All-Zero is a synthetic trivial strategy that makes the contaminated model fail every question (per-question solve probability identically 0).

5.2.2 Analysis: Sources of the Reversal

As shown in Figure 2, we test the reproducibility of the readout itself by drawing two independent sampling batches from the same M_{cl}. Under the discrete readout ($r^{0/1}$), the mean per-question gap between the two batches reaches 0.190, while under the solve-probability readout the two batches collapse onto one curve and the mean per-question gap drops to 0.041: the probability readout is far more stable per question than the discrete readout.

As shown in Table 2, two obstacles remain after the readout is corrected. The first is **cancellation**. LNE-blocking's over-suppression component Δ^-

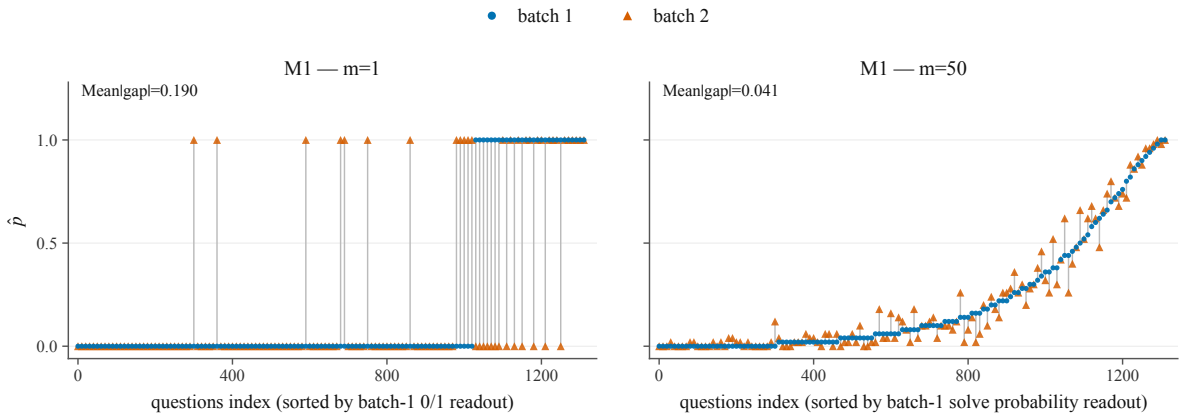


Figure 2: Per-question readouts of two independent sampling batches drawn from the same clean model M_{cl} (Llama-2, GSM8K domain). The horizontal axis is the question index sorted by the first batch's readout, and the vertical axis is the readout of each batch. Left: the discrete readout $r^{0/1}$, one sample per batch ($m = 1$, a small vertical jitter is added to avoid overlap). Right: the solve probability estimated from $m = 50$ samples of each batch.

(0.0836) is 2.0 times RailCap's (0.0420), yet its G-APP reading (0.0207) is 3.8 times better than RailCap's (0.0794): the extra collateral damage cancels against the residual contamination and pushes the reading closer to zero. The false perfection in Table 1 is not an artefact of estimation noise: under probability estimates, the G-APP reading is even closer to zero than Table 1's single-observation G-AP. With the readout held fixed, changing only the aggregation moves the verdict on LNE-blocking by a factor of 14 (0.0207 under G-APP versus 0.2932 under SA-PPG). A-PPG removes this cancellation (LNE-blocking 0.1879, above RailCap's 0.1634). The second is **the failure of equal weighting**. The All-Zero row drives every question's solve probability to zero (the limiting form of the trivial strategy described in §3.3). It reads 0.2190 under the equal-weight A-PPG, better than Identity (0.3793), TED (0.3536), and Shortcut (0.2399), whereas SA-PPG, after grouping by the clean model's solve probability, ranks it worst of all strategies (0.4903). The readout correction (§3.2.1), per-question differencing (§3.2.2), and stratified aggregation (§3.3) are each indispensable. SA-PPG combines the three into one metric.

	GSM8K			PQ		
Method	Llama2	Gemma	Pythia	Llama2	Gemma	Pythia
Identity	0.3261	0.3147	0.2131	0.2549	0.2205	0.1907
TED	0.3250	0.3049	0.1950	0.2524	0.2143	0.1903
Shortcut	0.2476	0.2395	0.2033	0.2242	0.2116	0.1708
LNE-blocking	0.2932	0.3061	0.2079	0.3046	0.2703	0.2171
RailCap	**0.1914**	**0.2313**	**0.1648**	**0.2090**	**0.1695**	**0.1429**

Table 3: SA-PPG of each mitigation strategy across the six settings (two contamination domains × three models). Every setting has $N = 1319$ questions, with the same-domain clean model M_{cl} as the reference. Bold marks the best strategy in each column.

5.3 Strategy Experiments: RailCap versus Prior Strategies

5.3.1 Main Result: Lowest SA-PPG across Settings

As shown in Table 3, RailCap attains the lowest SA-PPG in all six settings (two contamination domains by three models). Its largest lead is on Llama-2 × GSM8K (0.1914, against 0.2476 for the runner-up Shortcut). The baselines behave consistently across settings. TED is nearly indistinguishable from Identity. Even its largest gap, on Pythia × GSM8K, only moves 0.2131 to 0.1950. Shortcut is the runner-up in five of the six settings, a consistent but modest improvement. LNE-blocking trails both Shortcut and RailCap in all six settings: slightly better than Identity in the GSM8K domain, yet worse than Identity on all three models in the PQ domain. The contrast between the two domains suggests an explanation: in PQ, the questions seen at inference differ from the contaminated ones, so a one-shot estimate of the contamination made before operating becomes harder. The difficulty is not specific to LNE-blocking: Shortcut's lead over Identity narrows visibly on Llama-2 and Gemma (from 0.2476 versus 0.3261 to 0.2242 versus 0.2549 on Llama-2), and TED stays close to Identity in both domains. RailCap decides the amount of intervention step by step during genera-

Config	S-Δ_s^+	S-Δ_s^-	SA-PPG
Identity	0.2624	0.0637	0.3261
n=1	0.0215	0.2218	0.2434
n=3	0.0577	0.1412	0.1990
n=4	0.0942	**0.0972**	**0.1914**
n=5	0.0750	0.1219	0.1969
n=7	0.0869	0.1092	0.1961
n=4, ban	0.0726	0.1464	0.2190
All-Zero	**0.0000**	0.4903	0.4903

Table 4: Ablation of RailCap on Llama-2 × GSM8K. The n rows sweep the n-gram threshold under the default suppression (cap at the runner-up). In the n=4, ban row the cap is replaced by a hard ban: the rail token's probability is set to zero and it can no longer be sampled (the suppression-strength contrast). All-Zero is the trivial-strategy reference of §3.3. S-Δ_s^+ and S-Δ_s^- denote the stratified under-suppression component (residual contamination) and the stratified over-suppression component (collateral damage): the positive and negative parts of the per-question gaps are split within the groups and averaged the same way as SA-PPG, so the two sum to SA-PPG.

tion, without a pre-hoc estimate, and remains the best in all settings.

5.3.2 Ablation: n-gram Threshold and Suppression Form

As shown in Table 4, we ablate the two design choices of RailCap on Llama-2 × GSM8K: the n-gram threshold n and the form of suppression. Along the n axis, $n = 1$ triggers too frequently: the residual contamination nearly vanishes (S-Δ^+ 0.0215) while almost all the error comes from collateral damage (S-Δ^- 0.2218). As n grows, the triggering becomes more conservative, the residual component rises and the damage component falls, and $n = 4$ brings the two close to parity (0.0942 versus 0.0972) with the best SA-PPG (0.1914). Values of n from 3 to 7 all stay within 0.008 of the best, so the choice of n is robust. For the form of suppression, replacing the cap at the runner-up with a hard ban (n=4, ban) raises collateral damage to 0.1464 and SA-PPG to 0.2190: keeping the rail token available at reduced probability is preferable to prohibiting it entirely.

6 Conclusion

We studied two coupled problems in contamination mitigation evaluation: how to measure the restoration that a mitigation strategy achieves, and how to design a better strategy. On the metric side, we identified two flaws of the prevailing G-AP (a discrete readout that cannot represent per-question performance, and averaging before differencing that lets over- and under-suppression cancel out) and corrected them at the level of per-question solve probabilities. To keep strategies from pushing solve probabilities onto the clean model's frequent values under equal weighting, we further aggregate within groups of the clean model's solve probability, yielding SA-PPG. On the strategy side, we proposed RailCap, which checks at every decoding step whether a sample has fallen back onto the greedy trajectory and, once it has, caps the probability of the next trajectory token to that of the runner-up, turning the judgment of contamination from a one-shot pre-hoc estimate into stepwise supervision during generation. Across three model families and two forms of contamination, G-AP systematically overestimates the restoration of prior strategies, and RailCap attains the lowest SA-PPG in every setting.

References

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2023. Benchmarking foundation models with language-model-as-an-examiner. *Advances in Neural Information Processing Systems*, 36:78142–78167.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, St. Julian's, Malta. Association for Computational Linguistics.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large

language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

DeepSeek-AI. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*.

Chunyuan Deng, Yilun Zhao, Yuzhao Heng, Yitong Li, Jiannan Cao, Xiangru Tang, and Arman Cohan. 2024. Unveiling the spectrum of data contamination in language model: A survey from detection to remediation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16078–16092, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. *arXiv preprint arXiv:2104.08758*.

Ricardo Dominguez-Olmedo, Florian E. Dorner, and Moritz Hardt. 2025. Training on the test task confounds evaluation and emergence. In *The Thirteenth International Conference on Learning Representations*.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 12039–12050, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, Hannaneh Hajishirzi, Noah A. Smith, and Jesse Dodge. 2024. What's in my big data? In *The Twelfth International Conference on Learning Representations*.

Gemma Team. 2026. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*.

Shahriar Golchin and Mihai Surdeanu. 2024. Time travel in LLMs: Tracing data contamination in large language models. In *The Twelfth International Conference on Learning Representations*.

Ruijie Hou, Yueyang Jiao, Hanxu Hu, Yingming Li, Wai Lam, Huajian Zhang, and Hongyuan Lu. 2025. LNE-blocking: An efficient framework for contamination mitigation evaluation on large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3512–3528, Suzhou, China. Association for Computational Linguistics.

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. 2023. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. *arXiv preprint arXiv:2305.10160*.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Xiang Li, Yunshi Lan, and Chao Yang. 2025. Treeeval: Benchmark-free evaluation of large language models through tree planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24485–24493.

Yucheng Li. 2023. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation. *arXiv preprint arXiv:2309.10677*.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and Teknium. 2023. OpenOrca: An open dataset of GPT augmented FLAN reasoning traces. https://huggingface.co/datasets/Open-Orca/OpenOrca.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2024. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Wentao Ye, Jiaqi Hu, Liyao Li, Haobo Wang, Gang Chen, and Junbo Zhao. 2024. Data contamination

calibration for black-box LLMs. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10845–10861, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Wei Ye, Jindong Wang, Xing Xie, Yue Zhang, and Shikun Zhang. 2024. KIEval: A knowledge-grounded interactive evaluation framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5967–5985, Bangkok, Thailand. Association for Computational Linguistics.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, et al. 2024a. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint arXiv:2405.00332*.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024b. Pretraining Data Detection for Large Language Models: A Divergence-based Calibration Method. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5263–5274, Miami, Florida, USA. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. Don't make your llm an evaluation benchmark cheater. *arXiv preprint arXiv:2311.01964*.

Kejian Zhu, Shangqing Tu, Zhuoran Jin, Lei Hou, Juanzi Li, and Jun Zhao. 2025. Establishing trustworthy LLM evaluation via shortcut neuron analysis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3809–3822, Vienna, Austria. Association for Computational Linguistics.

Wenhong Zhu, Hongkun Hao, Zhiwei He, Yun-Ze Song, Jiao Yueyang, Yumeng Zhang, Hanxu Hu, Yiran Wei, Rui Wang, and Hongyuan Lu. 2024. CLEAN–EVAL: Clean evaluation on contaminated large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 835–847, Mexico City, Mexico. Association for Computational Linguistics.

A PQ Construction and Evaluation Details

A.1 PQ construction. For each leaked question, the question text is rewritten with DeepSeek-V4-Flash (DeepSeek-AI, 2026): the instruction requires a substantial rephrasing, keeps all numbers and the final answer identical, forbids any solution, and outputs exactly one line containing the rewritten question. The system prompt is:

> Significantly rephrase the given GSM8K math question. Keep ALL numbers and the final answer identical. Do not include any solution. Output exactly one line in this format and nothing else:
> New Question: <rephrased question>

The reasoning chain and the gold answer remain the GSM8K originals.