When Benchmarks Leak: Inference-Time Decontamination for LLMs

Chai Jianzhe[1] **Yu Zhe**[2] **Jun Sakuma**[1,2*]
[1]Institute of Science Tokyo
[2]RIKEN Center for Advanced Intelligence Project (AIP)
chai.j.4b1c@m.isct.ac.jp zhe.yu@riken.jp sakuma@c.titech.ac.jp

Abstract

Benchmark-based evaluation is the de facto standard for comparing large language models (LLMs). However, its reliability is increasingly threatened by test set contamination, where test samples or their close variants leak into training data and artificially inflate reported performance. To address this issue, prior work has explored two main lines of mitigation. One line attempts to identify and remove contaminated benchmark items before evaluation, but this inevitably alters the evaluation set itself and becomes unreliable when contamination is moderate or severe. The other line preserves the benchmark and instead suppresses contaminated behavior at evaluation time; however, such interventions often interfere with normal inference and lead to noticeable performance degradation on clean inputs. We propose DeconIEP, a decontamination framework that operates entirely during evaluation by applying small, bounded perturbations in the input embedding space. Guided by a relatively less-contaminated reference model, DeconIEP learns an instance-adaptive perturbation generator that steers the evaluated model away from memorization-driven shortcut pathways. Across multiple open-weight LLMs and benchmarks, extensive empirical results show that DeconIEP achieves strong decontamination effectiveness while incurring only minimal degradation in benign utility.

1 Introduction

Large language models (LLMs) have reached a level of performance that enables them to solve a broad range of knowledge, reasoning, and code-related tasks (Liang et al., 2023). As these models are increasingly developed, released, and deployed by different organizations, a systematic way to evaluate and compare their capabilities becomes necessary. In this setting, benchmark-based evaluation has naturally emerged as a widely adopted practice for assessing LLM performance. For example, prominent models such as GPT-5 (OpenAI, 2025) are often presented alongside their scores on established benchmarks like MMLU (Hendrycks et al., 2021a), where benchmark results serve as standardized evidence of general reasoning ability.

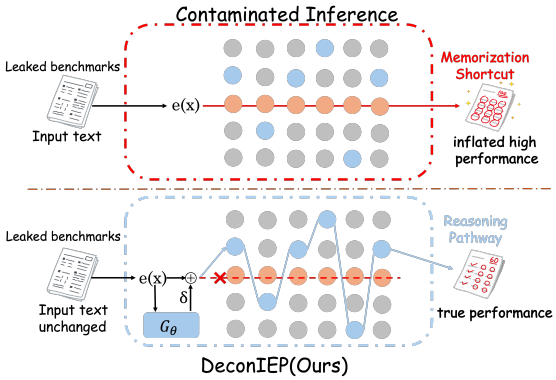


Figure 1: Overview: DeconIEP keeps the prompt fixed and adds a bounded embedding perturbation $\delta = G_\theta(e(x))$ to suppress memorization shortcuts on leaked samples, steering inference toward reasoning and recovering clean performance.

Recently, the need for trustworthy evaluation is further amplified by the rapid growth of *open-weight* LLMs. Unlike closed APIs, open-weight models can be freely downloaded, fine-tuned, merged, and re-distributed, enabling fast iteration by both industry and the open-source community. As a result, benchmark scores are increasingly used not only for marketing, but also for model selection, deployment decisions, and reproducible scientific comparison across independently trained variants. This makes reliable evaluation of open-weight models particularly important: their training pipelines and downstream fine-tuning are more decentralized, and thus more prone to unintended benchmark exposure and evaluation artifacts.

However, benchmark-centric evaluation is vul-

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 44743–44760
July 2-7, 2026 ©2026 Association for Computational Linguistics

nerable to data contamination (Deng et al., 2024a,b; Chen et al., 2025), where test items (or closely related variants) appear in training data thereby violating the train–test separation and inflating reported performance. This inflation can misrepresent real-world generalization. For example, it has been reported that over 15 LLMs exhibit inflated benchmark performance on six popular benchmarks, where the estimated contamination rate ranges from 1% to 45%, and some of them are fine-tuned models derived from open-weight models (Li et al., 2024).

To mitigate contamination, prior work has explored multiple strategies, including detect-then-filter approaches based on membership inference attacks (MIA) (Salem et al., 2018; Hayes et al., 2018). These methods apply an MIA detector to identify leaked benchmark items and exclude them from evaluation. However, filtering contaminated samples fundamentally alters the benchmark, making the resulting evaluation set no longer directly comparable to the original benchmark used in prior work. Moreover, MIA detectors are inherently imperfect: any non-zero false-negative rate leaves residual leaked items unfiltered, which can continue to inflate reported performance, especially under non-trivial contamination. As shown in Appendix B, we formally prove this limitation and validate it empirically.

These limitations motivate inference-time decontamination (Zhu et al., 2024; Dong et al., 2024; Zhu et al., 2025), which mitigates contamination effects during model inference rather than by filtering benchmark items. A common issue of existing methods is that the same intervention used for decontamination applied to contaminated inputs also affects clean ones: interventions strong enough to reduce contamination often alter model behavior on uncontaminated inputs, leading to non-trivial drops in clean accuracy. For example, rewriting methods (Zhu et al., 2024) may shift the effective input distribution, while internal-intervention approaches (Zhu et al., 2025; Menta et al., 2025) can suppress internal signals that are useful for both contaminated and clean examples. As a result, existing methods face an unfavorable trade-off between contamination mitigation and clean performance, motivating our goal of minimizing benign utility degradation.

To address this limitation, we propose a decontamination framework, termed DeconIEP (**Decon**tamination via **I**nference-time **E**mbedding **P**erturbation). We observe that data contamination mainly manifests as a change in how a model internally supports its predictions. On clean test samples, predictions are typically supported by broadly shared reasoning features that generalize across many inputs. In contrast, as shown in the first row of Figure 1, on contaminated test samples, the same outputs can be produced by relying on a different set of contamination-specific components, such as shortcut neurons or late-layer retrieval pathways, thereby reducing the need for genuine reasoning (Zhu et al., 2025; Menta et al., 2025). This observation motivates our approach: as shown in the second row of Figure 1, by applying small, targeted perturbations in the embedding space, we weaken contamination-driven dependencies and steer the model toward a cleaner inference regime, while preserving the input representation.

Based on this insight, we train a generator to produce inference-time embedding perturbations, guided by a comparatively less contaminated reference model. Our setting targets models obtained via fine-tuning from a known pretrained checkpoint (e.g., Llama3), where benchmark contamination is typically introduced or amplified by fine-tuning the checkpoint. In this context, an earlier checkpoint or base model with the same architecture naturally provides a practical reference with reduced benchmark-specific exposure. Importantly, our empirical evaluation demonstrates that our approach does not require a perfectly clean reference model. It suffices that the reference exhibits relatively less contamination, allowing the generator to suppress contamination-driven shortcuts while preserving benign reasoning behavior, without modifying the parameters of the evaluated model.

Contributions Our contributions are two-fold: (1) We propose DeconIEP, a novel inference-time decontamination method that mitigates benchmark contamination by applying small, targeted perturbations to input embeddings, without modifying the model parameters or the benchmark itself. (2) We conduct extensive experiments across multiple benchmarks and contamination settings, showing that DeconIEP effectively reduces contamination-induced performance inflation while incurring substantially smaller degradation on clean inputs compared to the existing inference-time baselines.

Method	Setting	Core Mechanism	Need Reference Model	Decontam. Efficacy ↑	Clean Eval Drop ↓
TED (Dong et al., 2024)	Black-box	Distribution calibration	No	Medium	Medium
ITD (Zhu et al., 2024)	Black-box	Query rewriting	No	Limited	**Low**
Shortcut Neuron (Zhu et al., 2025)	White-box	Activation patching	**Yes**	**High**	High
Short-circuiting (Menta et al., 2025)	White-box	Attention bypassing	No	**High**	High
DeconIEP(Ours)	**White-box**	**Embedding perturbation**	**Yes**	**High**	**Low**

Table 1: Comparison of decontamination strategies. *Decontam. Efficacy* indicates decontamination effectiveness on contaminated benchmarks. *Clean Eval Drop* indicates performance degradation on non-contaminated tasks. *Need Reference Model* indicates whether a reference model is required.

2 Related Work

Benchmark Contamination and Detection Data contamination (test-set leakage) in widely used benchmarks (e.g., MMLU, GSM8K) can inflate reported performance by inducing memorization rather than genuine generalization (Chen et al., 2025; Deng et al., 2024b; Xu et al., 2024). When training data are accessible, overlap-based checks can reveal contamination (Brown et al., 2020; Gao et al., 2021), but for most open-source models, such verification is infeasible. An alternative line of work adopts detect-then-filter strategies based on membership inference attacks (MIAs), but imperfect detectors leave residual contamination and may introduce selection bias (see Appendix B).

Inference-time Decontamination (Black-box) To avoid training-data access, API-only methods intervene at inference time: TED (Dong et al., 2024) calibrates outputs via repeated sampling, and ITD (Zhu et al., 2024) rewrites prompts using auxiliary LLMs. These approaches are easy to deploy but can be computationally expensive, and prompt rewriting may alter semantics or task difficulty (Xu et al., 2024).

Inference-time Decontamination (White-box) With open-weight access, prior work manipulates internal representations, Short-cut Neuron Analysis (Zhu et al., 2025) uses a reference model to identify such shortcut neurons and then edits their activations at test time. As shown in Table 1, it can achieve high decontamination efficacy, but often leads to a large clean-evaluation drop when the edited neurons overlap with general-purpose computation. Menta et al. (Menta et al., 2025) propose attention short-circuiting, which replaces the attention mixing operation with an identity mapping so that value vectors are forwarded without cross-token aggregation. They report that bypassing attention in deeper layers can substantially reduce the generation of memorized content. While this mechanism is not proposed as a decontamination method, it can be repurposed to suppress contamination-driven memorization. However, as summarized in Table 1, the same intervention also degrades general capabilities, leading to a large benign utility drop.

3 Problem Setup

Model Let $f : \mathcal{X} \to \mathcal{Y}$ be a Large Language Model (LLM), where $\mathcal{X}$ is the space of input sequences and $\mathcal{Y}$ is the space of output sequences.

Data Contamination We want to train a model f and we have a test benchmark $D_{\text{test}} \sim \mathcal{P}_{\text{test}}$ to evaluate its performance. We define the *contaminated* model f_{con} as any model trained on a dataset

$$D_{\text{train}}^{\text{con}} = D_{\text{train}}^{\text{clean}} \cup D_{\text{con}},$$

where $D_{\text{train}}^{\text{clean}}$ is general training data and unrelated to D_{test}, and D_{con} potentially causes training data contamination, where we consider the following three levels of contamination as follows.

1. **Exact Contamination**: $D_{\text{con}} \subseteq D_{\text{test}}$, i.e., the training data contains exact copies of test examples.

2. **Semantic-level Contamination**: For each $x \in D_{\text{test}}$, there exists $x' \in D_{\text{con}}$ such that, $\text{sem}(x) = \text{sem}(x')$. Here, $\text{sem}(\cdot)$ refers to paraphrasing or semantically equivalent rewriting by LLM or a human.

3. **Domain-level Contamination**: $D_{\text{con}} \sim \mathcal{P}_{\text{test}}$, i.e., the contamination set and the test set are drawn from (approximately) the same underlying distribution, such as D_{con} and D_{test} are two random splits of the same benchmark.

For a fixed test benchmark D_{test}, we call a model f_{clean} trained on $D_{\text{train}}^{\text{clean}}$ *clean* if its training data

does not contaminate D_{test} at any of three contamination levels defined above.

Due to contamination, the observed performance of f_{con} on D_{test} is often inflated:

$$\text{Perf}(f_{\text{con}}, D_{\text{test}}) > \text{Perf}(f_{\text{clean}}, D_{\text{test}}) \quad (1)$$

Here, $\text{Perf}(\cdot)$ is a general performance evaluator, for example, classification accuracy for multiple-choice benchmarks (e.g., MMLU).

Objective We design an inference-time mitigation operator

$$M : \mathcal{F} \to \mathcal{F}$$

which maps a contaminated model f_{con} to a mitigated predictor $M(f_{\text{con}})$ with a modified inference procedure. Our objective is to make its test performance match that of an ideal clean model:

$$\text{Perf}\big(M(f_{\text{con}}), D_{\text{test}}\big) \approx \text{Perf}\big(f_{\text{clean}}, D_{\text{test}}\big), \quad (2)$$

4 Methodology

4.1 Overview

We reduce contamination-induced inflation by correcting inference rather than filtering items. Given f_{con} and input x, we keep the prompt unchanged and add a bounded embedding perturbation $\delta(x)$ with $\|\delta(x)\|_\infty \le \zeta$. We train a generator G_θ via a KL objective (Theorem 1) to align f_{con} with a reference model f_{ref}. At test time, we apply $\delta(x)$ once per input and evaluate on the original benchmark.

4.2 KL Divergence as a Surrogate Objective

Our goal is to align the behavior of the mitigated contaminated model $M(f_{\text{con}})$ with the ideal clean model f_{clean} on the test benchmark D_{test}. We write the performance gap Δ_{perf} as:

$$\Delta_{\text{perf}} := \big|\text{Perf}(M(f_{\text{con}}), D_{\text{test}}) - \text{Perf}(f_{\text{clean}}, D_{\text{test}})\big|.$$

Directly minimizing Δ_{perf} is intractable because standard metrics (e.g., multiple-choice accuracy) are discrete and non-differentiable. We therefore adopt a differentiable, distribution-level surrogate. For each $x \in D_{\text{test}}$, let $p_{\text{con}}(\cdot \mid x)$ and $p_{\text{clean}}(\cdot \mid x)$ denote the output distributions of $M(f_{\text{con}})$ and f_{clean}, respectively. By viewing performance as $\mathbb{E}y \sim p(\cdot \mid x)[u(x, y)]$ for a bounded utility $u(x, y) \in [0, 1]$, the performance gap can be related to a distance between p_{con} and p_{clean}, requiring only boundedness and thus covering a range of evaluation metrics.

Theorem in Appendix C shows that the average KL divergence between the mitigated contaminated model and the clean model upper-bounds the performance gap. We therefore use KL minimization as a principled surrogate objective. This bound is only used as motivation and does not guarantee our algorithm's behavior. Next, we instantiate M so that the KL objective can be optimized efficiently.

4.3 Embedding-Space Perturbation for Inference-Time Mitigation

We implement the mitigation mechanism M by applying small, controlled perturbations in the embedding space, while keeping the discrete input text unchanged. This design preserves the semantic and difficulty invariance of the benchmark items, which input-space transformations, such as paraphrasing or rewriting, may violate.

Formally, let $e(x) \in \mathbb{R}^{L \times d}$ denote the input embedding sequence for a benchmark question x. We define the mitigated prediction as

$$M(f)(x) = f(e(x) + \delta(x)), \quad \|\delta(x)\|_\infty \le \zeta, \quad (3)$$

where $\delta(x)$ is a bounded perturbation with a small budget ζ. Constraining $\delta(x)$ in an ℓ_∞-ball encourage that the perturbed embedding remains in a local neighborhood of $e(x)$, aiming to preserve the underlying semantics and difficulty while disrupting memorized surface patterns.

This reasoning leads to the ideal per-sample objective of aligning the perturbed contaminated model with clean behavior:

$$\min_{\{\delta(x)\}_{x \in D}} \frac{1}{|D|} \sum_{x \in D} \text{KL}\big(f_{\text{con}}(e(x) + \delta(x)) \,\|\, f_{\text{clean}}(e(x))\big) \quad \text{s.t.} \quad \|\delta(x)\|_\infty \le \zeta, \ \forall x. \quad (4)$$

Directly optimizing a separate perturbation for each input is computationally infeasible at evaluation time. We therefore amortize the optimization by learning a generator G_θ. After training, we can use such a generator to generate $\delta(x)$ for all $x \in D_{\text{test}}$ (and other unseen inputs). Our amortized objective becomes:

$$\min_\theta \frac{1}{|D|} \sum_{x \in D} \text{KL}\Big(f_{\text{con}}\big(e(x) + G_\theta(e(x))\big) \,\|\, f_{\text{clean}}\big(e(x)\big)\Big) \quad \text{s.t.} \quad \big\|G_\theta(e(x))\big\|_\infty \le \zeta, \quad \forall x \in D. \quad (5)$$

where D is an auxiliary dataset used to train G_θ sampled from evaluated benchmark.

4.4 Generator Design

Setup We assume access to a contaminated model f_{con}, a reference model f_{ref}, and a small auxiliary dataset D_{aux} sampled from the benchmark. The reference model serves as a practical proxy for clean behavior during generator training. Since a strictly clean model f_{clean} is often unavailable, f_{ref} is typically chosen as the same-architecture base model (or an earlier checkpoint) from which f_{con} is fine-tuned, and is expected to have less benchmark-specific exposure. This choice is standard in contamination-related analyses (Zhu et al., 2025; Carlini et al., 2021, 2022).

Generator Architecture We parameterize the perturbation generator as a lightweight sequence-to-sequence network. Let G_θ denote a generator that maps the input embedding $e(x) \in \mathbb{R}^{L\times d}$ to a raw perturbation. To enforce the perturbation budget, we apply a scaled $\tanh$ transformation:

$$\delta(x) = \zeta \cdot \tanh(G_\theta(x)),$$

which guarantees $\|\delta(x)\|_\infty \le \zeta$ by construction. The perturbed embedding $e(x)+\delta(x)$ is then fed to f_{con} at inference time. Unless otherwise specified, G_θ is instantiated as a small decoder-only Transformer (Vaswani et al., 2017); architectural details are provided in Appendix D.

Training Objective For a training sample $x \sim D_{\text{aux}}$, let $p^{\text{con}} = f_{\text{con}}(e(x) + \delta(x))$ and $p^{\text{ref}} = f_{\text{ref}}(e(x))$ denote the output distributions of the contaminated and reference models, respectively. We train the generator using a composite KL+CE objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim D_{\text{aux}}}\Big[\lambda_{\text{KL}} \cdot \text{KL}\big(p^{\text{con}} \,\|\, p^{\text{ref}}\big) + \lambda_{\text{CE}} \cdot \text{CE}\big(p^{\text{con}}, y_{\text{ref}}\big)\Big], \tag{6}$$

where y_{ref} denotes hard token labels sampled from f_{ref}, and $\lambda_{\text{KL}}, \lambda_{\text{CE}}$ control the relative weights of the two terms. The KL term directly optimizes the surrogate objective motivated by Theorem 1, while the cross-entropy term stabilizes training and prevents degenerate solutions. In the appendix, Algorithm 1 summarizes the training procedure.

5 Experiment

5.1 Experiment Setup

Dataset We evaluate on two benchmarks: MMLU (Hendrycks et al., 2021b) and TruthfulQA (Lin et al., 2022). MMLU is a large-scale multiple-choice benchmark spanning diverse academic and professional domains, while TruthfulQA measures truthfulness and resistance to common misconceptions. In our experiments, these benchmarks serve dual roles. First, they define the evaluation benchmarks. Second, to simulate benchmark contamination, a small subset of benchmark items is intentionally allowed to leak into the training data when constructing contaminated models. All benchmark items used for contamination are drawn exclusively from the corresponding benchmark (MMLU or TruthfulQA). We also test the proposal on the code generation task, and show in the Appendix E.3.

Models We conduct experiments on three instruction-tuned large language models : LLaMA-3-8B-Instruct (Grattafiori et al., 2024), Qwen Family (Yang et al., 2025), and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023). These models represent different model families, allowing us to evaluate the generality of our decontamination method across heterogeneous architectures. Before simulating contamination, we start from the corresponding pretrained (unfine-tuned) model for each architecture. This pretrained model serves as the common initialization for all experiments and is also used as the reference model in our decontamination framework. Fine-tuning is then performed either with or without benchmark leakage, depending on the contamination setting. These model families span heterogeneous architectures, allowing us to evaluate the generality of our decontamination method.

Contamination simulation and evaluation Following common practice in prior work (Zhu et al., 2025), we simulate benchmark contamination in a controlled manner. For each benchmark, we construct a contaminated model f_{con} by fine-tuning the pretrained model on a mixture of: (i) a general instruction-tuning dataset OpenOrca, denoted as $D_{\text{train}}^{\text{clean}}$, and (ii) a small set of leaked benchmark items D_{con}, drawn from either MMLU or TruthfulQA. Each benchmark item in D_{con} is repeatedly included in training $o \in \{1, 3, 5\}$ times to simulate different contamination strengths. To isolate the effect of benchmark leakage, we also train a clean counterpart f_{clean} by fine-tuning on OpenOrca only, using the same total number of training samples as f_{con}, but without including any benchmark items.

We evaluate decontamination performance under three contamination settings, each associated with a different construction of the test set D_{test}:

Model	Split	Method	Access	TruthfulQA ($o = 3$)			MMLU ($o = 3$)		
				Exact	Semantic-level	Domain-level	Exact	Semantic-level	Domain-level
Mistral	f_{clean}	baseline	–	0.287	0.249	0.269	0.451	0.434	0.406
	f_{con}	W/O Decontamination	–	0.976 (0.689)	0.971 (0.722)	0.861 (0.592)	0.892 (0.441)	0.875 (0.441)	0.475 (0.069)
		TED	Black-box	0.535 (0.248)	0.579 (0.330)	0.740 (0.471)	0.224 (0.227)	0.388 (0.046)	0.375 (0.031)
		ITD	Black-box	0.976 (0.689)	0.971 (0.722)	0.861 (0.592)	0.886 (0.435)	0.863 (0.429)	0.471 (0.065)
		Shortcut Neuron	White-box	0.668 (0.381)	0.634 (0.385)	0.636 (0.367)	0.380 (0.071)	0.386 (0.048)	0.239 (0.167)
		Short Circuit	White-box	0.933 (0.646)	0.914 (0.665)	0.793 (0.524)	0.833 (0.382)	0.808 (0.374)	0.478 (0.072)
		DeconIEP(Ours)	White-box	**0.533 (0.246)**	**0.551 (0.302)**	**0.565 (0.296)**	**0.458 (0.007)**	**0.406 (0.028)**	**0.417 (0.011)**
Qwen2.5	f_{clean}	baseline	–	0.555	0.531	0.572	0.670	0.671	0.673
	f_{con}	W/O Decontamination	–	0.976 (0.421)	0.952 (0.421)	0.885 (0.313)	0.905 (0.235)	0.872 (0.201)	0.744 (0.071)
		TED	Black-box	0.880 (0.325)	0.900 (0.369)	0.846 (0.274)	0.837 (0.167)	0.882 (0.211)	0.713 (0.040)
		ITD	Black-box	0.976 (0.421)	0.947 (0.416)	0.899 (0.327)	0.898 (0.228)	0.868 (0.197)	0.751 (0.078)
		Shortcut Neuron	White-box	0.684 (0.129)	**0.526 (0.005)**	0.394 (0.178)	0.416 (0.254)	0.420 (0.251)	0.371 (0.302)
		Short Circuit	White-box	0.650 (0.095)	0.622 (0.091)	**0.614 (0.042)**	0.807 (0.137)	0.757 (0.086)	0.640 (0.033)
		DeconIEP(Ours)	White-box	**0.620 (0.065)**	0.632 (0.101)	0.641 (0.069)	**0.785 (0.115)**	**0.632 (0.039)**	**0.646 (0.027)**
LLaMA-3	f_{clean}	baseline	–	0.474	0.467	0.452	0.576	0.565	0.549
	f_{con}	W/O Decontamination	–	0.986 (0.512)	0.981 (0.514)	0.928 (0.476)	0.921 (0.345)	0.901 (0.336)	0.669 (0.120)
		TED	Black-box	0.966 (0.493)	0.951 (0.485)	0.923 (0.471)	0.864 (0.288)	0.842 (0.277)	0.638 (0.089)
		ITD	Black-box	0.943 (0.469)	0.957 (0.490)	0.899 (0.447)	0.886 (0.310)	0.844 (0.279)	0.648 (0.098)
		Shortcut Neuron	White-box	0.818 (0.345)	0.852 (0.385)	0.726 (0.274)	**0.690 (0.114)**	0.682 (0.117)	0.482 (0.068)
		Short Circuit	White-box	0.593 (0.120)	0.608 (0.141)	0.625 (0.173)	0.701 (0.125)	0.680 (0.115)	0.707 (0.157)
		DeconIEP(Ours)	White-box	**0.536 (0.062)**	**0.519 (0.053)**	**0.553 (0.101)**	0.741 (0.165)	**0.537 (0.028)**	**0.536 (0.014)**

Table 2: Decontamination results for $o = 3$ across different models. Values denote accuracy, with parentheses showing RC (smaller is better). Best RC is in bold, and the second best is underlined within each model and split. ($\zeta = 10^{-3}$)

(i) *Exact contamination:* D_{test} consists of a subset of the leaked benchmark items from D_{con}, evaluating direct memorization effects; (ii) *Semantic-level contamination:* D_{test} consists of paraphrases of the leaked benchmark items, which are semantically equivalent but not observed during training, with ground-truth answers unchanged; and (iii) *Domain-level contamination:* D_{test} is sampled from the same benchmark distribution but is strictly disjoint from D_{con}, evaluating same-domain generalization rather than direct leakage. For benign (clean) utility evaluation, we additionally evaluate on an external benchmark that is not used to construct D_{con}. Specifically, when contamination is simulated on MMLU, TruthfulQA is used as the clean test benchmark, and vice versa.

Implementation Details The noise generator G_θ is a 4-layer decoder-only Transformer ($n = 4$). The generator is trained on the same benchmark used to construct the contamination set (e.g., MMLU for MMLU-contaminated models), using benchmark samples that are disjoint from the test set in each evaluation setting. We set the perturbation budget $\zeta = 10^{-3}$, learning rate $lr = 1 \times 10^{-5}$, dropout rate to 0.2, and use an auxiliary dataset of size $|D_{\text{aux}}| = 400$, which is sampled from contaminated benchmark and disjoint from D_{test}. Additional details shown in Appendix E.

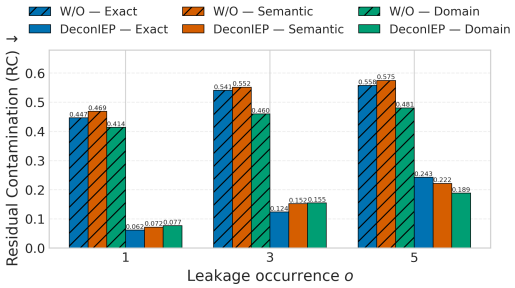
Metric We evaluate decontamination using two metrics: Residual Contamination (RC) and Benign Utility Drop (BUD). RC measures how close the decontaminated model $M(f_{\text{con}})$ is to the uncontaminated model f_{clean} on contaminated data, while BUD measures utility loss on clean data.

Smaller values indicate better decontamination. Formal definition see AppendixE.1.

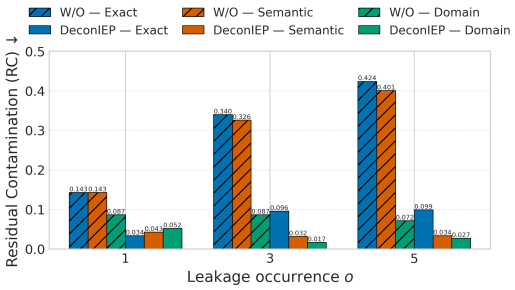
Comparison methods We compare our method with representative inference-time decontamination baselines explained in Section 2: TED and ITD as black-box methods, Shortcut Neuron and Short Circuit as white-box methods.

5.2 Evaluation of Decontamination Effectiveness

We first evaluate decontamination by whether the performance of the post-mitigation model approaches that of the uncontaminated model f_{clean}. Table 2 reports accuracy and Residual Contamination (RC; lower is better) on TruthfulQA and MMLU under $o = 3$, $o = 1$ and $o = 5$ see Appendix E.2. Overall, DeconIEP achieves the lowest or near-lowest RC across models, benchmarks, and contamination granularities, and remains robust under three-level contamination. While some base-

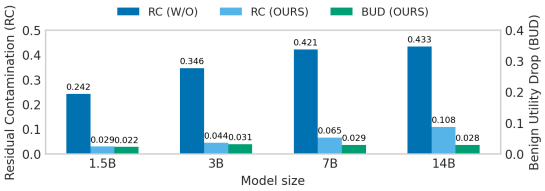


(a) TruthfulQA.

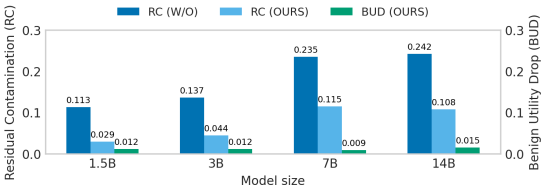


(b) MMLU.

Figure 2: Average Residual Contamination (RC) of all models across leakage occurrences o under different contamination levels (lower is better).($\zeta = 10^{-3}$)



(a) TruthfulQA ($o = 3$).



(b) MMLU ($o = 3$).

Figure 3: Scaling behavior on Qwen2.5-Instruct models ($o = 3$). We compare RC on exact domain and BUD across model sizes (1.5B/3B/7B/14B) on (a) TruthfulQA and (b) MMLU. Our method remains effective and stable under scaling with a fixed embedding perturbation budget $\zeta = 10^{-3}$.

lines are best in isolated cases, their improvements are not consistent.

Figure 2 further varies the occurrence level o. Although larger o increases residuals for the unmitigated model, DeconIEP consistently reduces RC and keeps accuracy closer to f_{clean}. To assess whether decontamination introduces unintended degradation on clean tasks, we evaluate *benign utility* using a cross-benchmark clean evaluation. Specifically, to isolate benign utility from contamination-related effects, models contaminated on MMLU are evaluated on the unrelated TruthfulQA benchmark, and vice versa. As shown in Table 3, DeconIEP incurs minimal BUD across all model families, whereas several white-box baselines exhibit substantially larger utility degradation. These results indicate that the proposed embedding-space intervention effectively mitigates contamination while preserving benign performance, with advantages that remain stable across heterogeneous architectures.

5.3 Evaluation across Model scale

We evaluate our decontamination method across model scales using four Qwen2.5 models (Yang et al., 2025): Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-14B-Instruct. Figure 3 shows that our method maintains stable decontamination effectiveness across all scales, indicating that it generalizes beyond a specific model size, with only mild variation across model capacities.

5.4 Semantic preservation

Then, we verify that embedding perturbations preserve input semantics. We measure the average cosine similarity between the original embedding and the perturbed embedding,

$$\text{Cos}(\zeta) \;=\; \mathbb{E}_x\Big[\cos\big(e(x),\, e(x)+\delta(x)\big)\Big],$$

using the same pooling operator as in our implementation. Figure 4 shows that for practical budgets $\zeta \le 10^{-3}$, cosine similarity remains close to 1, and consistently exceeds the similarity between the original prompt and its rewritten counterpart. This suggests that embedding-level intervention introduces less semantic drift than rewriting while still enabling effective decontamination.

5.5 Ablation Study

Controllability via the perturbation budget ζ We next treat ζ as an explicit control knob that trades off decontamination strength against benign utility. Table 4 shows the relationship between RC and BUD induced by varying ζ. As ζ increases, DeconIEP moves toward lower RC at the cost of

Model	Method	Access	Benign Utility Drop ↓	
			TruthfulQA	MMLU
Mistral	DeconIEP	White-box	0.019	0.016
	TED	Black-box	0.028	0.030
	ITD	Black-box	0.000	0.001
	Shortcut Neuron	White-box	0.129	0.152
	Short Circuit	White-box	0.057	0.077
Qwen2.5	DeconIEP	White-box	0.009	0.029
	TED	Black-box	0.110	0.022
	ITD	Black-box	0.000	0.002
	Shortcut Neuron	White-box	0.207	0.256
	Short Circuit	White-box	0.172	0.238
LLaMA-3	DeconIEP	White-box	0.031	0.041
	TED	Black-box	0.012	0.035
	ITD	Black-box	0.010	0.016
	Shortcut Neuron	White-box	0.057	0.190
	Short Circuit	White-box	0.105	0.161

Table 3: BUD under cross-benchmark evaluation ($o = 3$). We contaminate each model on MMLU and measure the utility drop on TruthfulQA to assess benign performance on clean data, and vice versa. ($\zeta = 10^{-3}$)

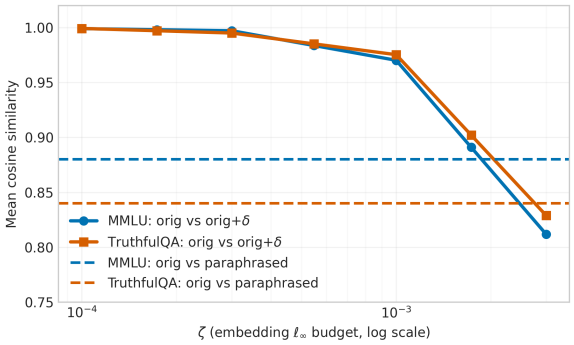


Figure 4: Semantic invariance under embedding perturbations. Mean cosine similarity between original and perturbed embeddings (orig vs. orig+δ) versus ζ on MMLU and TruthfulQA. Dashed lines show orig vs. paraphrased as a semantic-preserving reference.

higher BUD, forming a clear and monotonic trade-off curve. Compared with representative baselines (see Appendix E.4), DeconIEP provides a more favorable RC–BUD region, and the practical budget regime yields strong contamination suppression with limited utility loss.

Effect of reference model Finally, we examine the role of the reference model. While our method uses a reference model to guide perturbation generation, a perfectly clean anchor may be unavailable in practice. We therefore test robustness when the reference model is mildly contaminated.

We construct imperfect references by fine-tuning the reference model on data containing different proportions of benchmark test items, yielding varying contamination ratios, and then run our method

ζ	TruthfulQA		MMLU	
	RC ↓	BUD ↓	RC ↓	BUD ↓
1e−5	0.512	0.000	0.344	0.000
3e−5	0.393	0.009	0.304	0.000
1e−4	0.226	0.024	0.214	0.022
3e−4	0.096	0.034	0.184	0.031
1e−3	0.062	0.038	0.165	0.041
3e−3	0.024	0.050	0.034	0.070
1e−2	0.022	0.062	0.032	0.102

Table 4: RC–BUD trade-off induced by the perturbation budget ζ. We report RC and BUD for DeconIEP under different ζ on TruthfulQA and MMLU for contaminated Llama 3 model, $o = 3$.

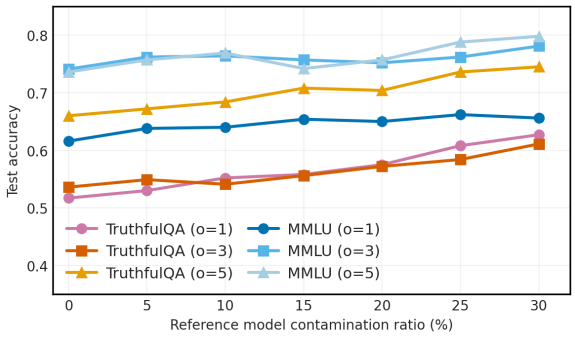


Figure 5: Test accuracy under increasing contamination in the reference model.

under the same contaminated evaluation setting. Figure 5 shows that performance is largely stable as the reference contamination increases from 0% to 30% across benchmarks and occurrence levels. This indicates that the reference need not be strictly clean: even mildly contaminated, it provides a sufficiently informative anchor distribution to steer the evaluated model away from contamination-driven behavior. In practice, a readily available same-architecture base model can serve as a reference.

We also do experiments to investigate the difference between our perturbation and random noise, how the sample size of D_{aux} affects our proposal; due to space constraints, we report these results in Appendix E.5 and E.6.

6 Conclusion

Test-set contamination inflates benchmark scores, and detect-then-filter evaluation fails under moderate contamination due to detector errors. We propose DeconIEP, an inference-time method that applies small, ℓ_∞-bounded perturbations to input embeddings to align a contaminated model with a less-contaminated reference. Across multiple open-

weight LLM families and two benchmarks, DeconIEP reduces residual contamination with minimal benign utility drop.

7 Limitations

Our study has several limitations. First, DeconIEP is a white-box method: it requires access to input embeddings (and, in our training setup, gradients) to generate and apply perturbations, which limits applicability to closed-model APIs without additional approximations. Second, DeconIEP relies on a reference model to provide a comparatively less-contaminated behavioral anchor. While we empirically observe robustness to moderate reference contamination, performance may degrade when the reference is heavily contaminated, distributionally mismatched, or differs substantially in alignment behavior from the evaluated model. Then, although our intervention is bounded and empirically exhibits strong semantic invariance (e.g., high cosine similarity under small ζ), we do not provide formal guarantees that semantics/difficulty are preserved for all inputs. Finally, DeconIEP introduces additional inference overhead for perturbation generation and reference-guided objectives; while lightweight relative to multi-query black-box baselines, reducing overhead for very large-scale evaluation remains an important direction.

Acknowledgements

This work was supported in part by JST under Grant Numbers JPMJNX25C2, JPMJKP24C3, JPMJCR23M4, and JPMJCR21D3, and by JSPS under Grant Numbers 23H00483 and JPJSBP120251002.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

Simin Chen, Yiming Chen, Zexin Li, Yifan Jiang, Zhongwei Wan, Yixin He, Dezhi Ran, Tianle Gu, Haizhou Li, Tao Xie, and 1 others. 2025. Benchmarking large language models under data contamination: A survey from static to dynamic evaluation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10091–10109.

Chunyuan Deng, Yilun Zhao, Yuzhao Heng, Yitong Li, Jiannan Cao, Xiangru Tang, and Arman Cohan. 2024a. Unveiling the spectrum of data contamination in language models: A survey from detection to remediation. *Preprint*, arXiv:2406.14644.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024b. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8706–8719.

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.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, and 1 others. 2021. A framework for few-shot language model evaluation. *Zenodo*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2018. Logan: Membership inference attacks against generative models. *Preprint*, arXiv:1705.07663.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Yucheng Li, Yunhao Guo, Frank Guerin, and Chenghua Lin. 2024. An open-source data contamination report for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 528–541, Miami, Florida, USA. Association for Computational Linguistics.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, and 31 others. 2023. Holistic evaluation of language models. *Preprint*, arXiv:2211.09110.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

Chris Liu, Yaxuan Wang, Jeffrey Flanigan, and Yang Liu. 2024. Large language model unlearning via embedding-corrupted prompts. *Advances in Neural Information Processing Systems*, 37:118198–118266.

Tarun Ram Menta, Susmit Agrawal, and Chirag Agarwal. 2025. Analyzing memorization in large language models through the lens of model attribution. *arXiv preprint arXiv:2501.05078*.

OpenAI. 2025. Introducing gpt-5. `https://openai.com/index/introducing-gpt-5/`. Online; accessed 2026-01-05.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *Preprint*, arXiv:1806.01246.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. Detecting pretraining data from large language models. *arXiv preprint arXiv:2310.16789*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Kejian Zhu, Shangqing Tu, Zhuoran Jin, Lei Hou, Juanzi Li, and Jun Zhao. 2025. Establishing trustworthy llm evaluation via shortcut neuron analysis. *Preprint*, arXiv:2506.04142.

Qin Zhu, Qinyuan Cheng, Runyu Peng, Xiaonan Li, Ru Peng, Tengxiao Liu, Xipeng Qiu, and Xuan-Jing Huang. 2024. Inference-time decontamination: Reusing leaked benchmarks for large language model evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9113–9129.

A Related Work

A.1 Data Contamination

Test-set leakage in widely used benchmarks (e.g., MMLU, GSM8K) has become pervasive, where inflated scores may reflect memorization rather than genuine generalization (Chen et al., 2025; Deng et al., 2024b; Xu et al., 2024). When an auditor has full access to the training corpus, contamination can be identified by direct overlap checks (exact match or n-gram similarity) between training documents and benchmark instances (Brown et al., 2020; Gao et al., 2021). In recent years, the distribution of open-source models via platforms like Hugging Face has become widespread, but information disclosure regarding their training data remains scarce, leaving no means to verify whether benchmarking is being conducted properly. A straightforward approach is to use MIA to detect whether test instances were seen during training and filter out detected contaminated items before evaluation. However, imperfect detection can leave residual contamination and bias the reported scores. This motivates *inference-time* approaches that aim to correct contaminated behavior without modifying model weights or relying on the training data.

A.2 Inference-time Decontamination

Black-box methods Early inference-time decontamination methods assume evaluators only have API access to the model, without access to internal parameters or activations. As summarized in Table 1, black-box approaches mainly rely on output-side interventions. TED (Dong et al., 2024) frames decontamination as distribution calibration:

it keeps model parameters fixed and approximates a "clean" output distribution via repeated stochastic decoding. ITD (Zhu et al., 2024) rewrites potentially contaminated prompts using an auxiliary model like GPT , aiming to reduce direct reuse of memorized patterns. These methods are convenient to deploy, but they typically incur high inference cost, including extra model calls (e.g., auxiliary LLM rewriting), repeated decoding/sampling, and can be limited against strongly memorized contamination; moreover, rewriting-based methods do not strictly preserve semantics and may change question difficulty (Xu et al., 2024).

White-box methods For benchmark evaluation, white-box access has become increasingly practical due to the availability of open-weight LLMs. White-box access enables targeted inference-time interventions using internal signals such as parameters, intermediate activations, and layer-wise representations. Recent work in this regime can be grouped into three directions (Table 1).

(i) Neuron/activation patching. This line localizes Neurons in LLM that drive contaminated behavior and patches their activations during inference. Shortcut Neuron Analysis (Zhu et al., 2025) uses a reference model to identify such shortcut neurons and then edits their activations at test time. As shown in Table 1, it can achieve high decontamination efficacy, but often leads to a large clean-evaluation drop when the edited neurons overlap with general-purpose computation.

(ii) Architectural bypassing to weaken memorization. A second line modifies specific modules in the forward pass to reduce memorization-driven behavior during decoding. Since contamination effects often manifest as answer retrieval on leaked items (Xu et al., 2024), weakening memorization at inference time can reduce contamination-induced score inflation. Menta et al. (Menta et al., 2025) propose attention short-circuiting by replacing attention mixing with an identity operation, so that values are forwarded without cross-token attention. They show that bypassing attention in deeper layers can substantially reduce memorized content generation. However, as summarized in Table 1, this approach will also cause a large clean-evaluation drop.

(iii) Inference-time unlearning. A third direction modifies the *input* of inference time to make the model infer the input in an "unlearned" state. Embedding-Corrupted (ECO) Prompts (Liu et al., 2024) train a token classifier to detect tokens to be unlearned, and then apply random noises in the embedding space of those tokens. The random noise is optimized offline via zeroth-order procedures, and the resulting embedding-corrupted prompts are applied only at inference time, requiring no changes to LLM weights. ECO has a low inference cost but typically offers limited decontamination efficacy. A key limitation is that the corruption is largely stochastic and provides limited directional control over memorized representations, thereby reducing effectiveness on strongly memorized items while still perturbing useful features on clean inputs.

Our approach Our method follows the white-box, inference-time paradigm, but intervenes exclusively at the input embedding level through a lightweight generator. Compared with activation patching, we avoid directly editing internal circuits, reducing the risk of over-suppressing general capabilities. Compared with coarse architectural bypassing, we provide a data-driven and instance-adaptive mechanism that targets contamination-induced behavior more selectively. Compared with ECO, DeconIEP learns instance-adaptive and directionally controlled perturbations, avoiding purely stochastic corruption and eliminating the need for a token classifier.

Overall, this design yields competitive mitigation of contamination while substantially reducing performance sacrifice on clean benchmarks and incurring only small inference overhead.

B Limitations of "Detect-then-Filter"

A widely considered mitigation approach is to first apply a membership inference algorithm (MIA) to the benchmark, and then evaluate only on examples predicted as "non-member." Let $D_{\text{test}} = D_{\text{clean}} \cup D_{\text{con}}$ with $D_{\text{clean}} \cap D_{\text{con}} = \emptyset$. We model an arbitrary MIA detector as a binary classifier $\text{MIA} : \mathcal{X} \to \{0, 1\}$:

$$\text{MIA}(x \in D_{\text{test}}) = \begin{cases} 0, & x \in D_{\text{clean}}, \quad (7) \\ 1, & x \in D_{\text{con}} . \quad (8) \end{cases}$$

For this detector, we define

$$\text{FNR} = \Pr[\text{MIA}(x) = 0 \mid x \in D_{\text{con}}],$$
$$\text{FPR} = \Pr[\text{MIA}(x) = 1 \mid x \in D_{\text{clean}}],$$

and retain after detection

$$D_{\text{neg}} = \{x \in D_{\text{test}} : \text{MIA}(x) = 0\},$$

on which evaluation is performed. We denote the contaminated model by f_{con}, the clean model by f_{clean}, and the contamination rate by $\alpha = |D_{\text{con}}|/|D_{\text{test}}|$.

A simple counting argument yields the expected performance of f_{con} on D_{neg}:

$$\text{Perf}(D_{\text{neg}}, f_{\text{con}}) = \frac{(1-\alpha)(1-\text{FPR})\,\text{Perf}(D_{\text{clean}}, f_{\text{con}}) + \alpha\,\text{FNR}\,\text{Perf}(D_{\text{con}}, f_{\text{con}})}{(1-\alpha)(1-\text{FPR}) + \alpha\,\text{FNR}} \tag{9}$$

Derivation of Eq. (9) We view performance as the average expected utility under the model output distribution.

$$\text{Perf}(S, f_{\text{con}}) = \frac{1}{|S|}\sum_{x\in S}\mathbb{E}_{y\sim p_{\text{con}}(\cdot|x)}\Big[u(x,y)\Big]. \tag{10}$$

We define the filtered (negative) subset retained by the detector.

$$D_{\text{neg}} = \Big\{x \in D_{\text{test}} : \text{MIA}(x) = 0\Big\}. \tag{11}$$

We parameterize the test set composition by the contamination rate and the dataset size.

$$\begin{aligned} N &= |D_{\text{test}}|, \\ \alpha &= \frac{|D_{\text{con}}|}{|D_{\text{test}}|}, \\ |D_{\text{clean}}| &= (1-\alpha)N, \\ |D_{\text{con}}| &= \alpha N. \end{aligned} \tag{12}$$

We recall the detector error rates.

$$\begin{aligned} \text{FNR} &= \Pr\big[\text{MIA}(x) = 0 \mid x \in D_{\text{con}}\big], \\ \text{FPR} &= \Pr\big[\text{MIA}(x) = 1 \mid x \in D_{\text{clean}}\big]. \end{aligned} \tag{13}$$

From the definitions above, the keep (negative) probabilities are

$$\begin{aligned} \Pr\big[x \in D_{\text{neg}} \mid x \in D_{\text{clean}}\big] &= \Pr\big[\text{MIA}(x) = 0 \mid x \in D_{\text{clean}}\big] \\ &= 1 - \text{FPR}, \\ \Pr\big[x \in D_{\text{neg}} \mid x \in D_{\text{con}}\big] &= \Pr\big[\text{MIA}(x) = 0 \mid x \in D_{\text{con}}\big] \\ &= \text{FNR}. \end{aligned} \tag{14}$$

Therefore, the expected counts of retained clean / contaminated examples are

$$\begin{aligned} \mathbb{E}\big[|D_{\text{neg}} \cap D_{\text{clean}}|\big] &= (1-\alpha)N(1-\text{FPR}), \\ \mathbb{E}\big[|D_{\text{neg}} \cap D_{\text{con}}|\big] &= \alpha N\,\text{FNR}. \end{aligned} \tag{15}$$

Summing yields the expected size of the filtered set.

$$\mathbb{E}\big[|D_{\text{neg}}|\big] = (1-\alpha)N(1-\text{FPR}) + \alpha N\,\text{FNR}. \tag{16}$$

Define the per-input expected utility under the contaminated model.

$$\bar{u}_{\text{con}}(x) = \mathbb{E}_{y\sim p_{\text{con}}(\cdot|x)}\Big[u(x,y)\Big]. \tag{17}$$

The total retained utility decomposes into two disjoint parts.

$$\sum_{x\in D_{\text{neg}}}\bar{u}_{\text{con}}(x) = \sum_{x\in D_{\text{neg}}\cap D_{\text{clean}}}\bar{u}_{\text{con}}(x) + \sum_{x\in D_{\text{neg}}\cap D_{\text{con}}}\bar{u}_{\text{con}}(x). \tag{18}$$

Assuming the detector induces *composition-only* selection within each split (i.e., retained examples are representative within D_{clean} and within D_{con}), we have

$$\begin{aligned} \mathbb{E}\left[\sum_{x\in D_{\text{neg}}\cap D_{\text{clean}}}\bar{u}_{\text{con}}(x)\right] &= \mathbb{E}\big[|D_{\text{neg}} \cap D_{\text{clean}}|\big]\,\text{Perf}(D_{\text{clean}}, f_{\text{con}}), \\ \mathbb{E}\left[\sum_{x\in D_{\text{neg}}\cap D_{\text{con}}}\bar{u}_{\text{con}}(x)\right] &= \mathbb{E}\big[|D_{\text{neg}} \cap D_{\text{con}}|\big]\,\text{Perf}(D_{\text{con}}, f_{\text{con}}). \end{aligned} \tag{19}$$

Substituting Eq. (15) gives the expected total retained utility.

$$\begin{aligned} \mathbb{E}\left[\sum_{x\in D_{\text{neg}}}\bar{u}_{\text{con}}(x)\right] &= (1-\alpha)N(1-\text{FPR})\,\text{Perf}(D_{\text{clean}}, f_{\text{con}}) \\ &\quad + \alpha N\,\text{FNR}\,\text{Perf}(D_{\text{con}}, f_{\text{con}}). \end{aligned} \tag{20}$$

By definition, the filtered performance is

$$\text{Perf}(D_{\text{neg}}, f_{\text{con}}) = \frac{1}{|D_{\text{neg}}|}\sum_{x\in D_{\text{neg}}}\bar{u}_{\text{con}}(x). \tag{21}$$

Using a large-N approximation (ratio of expectations), we obtain

$$\mathbb{E}\big[\text{Perf}(D_{\text{neg}}, f_{\text{con}})\big] \approx \frac{\mathbb{E}\Big[\sum_{x\in D_{\text{neg}}}\bar{u}_{\text{con}}(x)\Big]}{\mathbb{E}\big[|D_{\text{neg}}|\big]}. \tag{22}$$

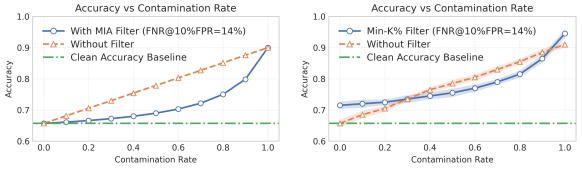
Plugging Eq. (20) and Eq. (16), and canceling the common factor N, yields

$$\mathbb{E}\big[\text{Perf}(D_{\text{neg}}, f_{\text{con}})\big] \approx \frac{(1-\alpha)(1-\text{FPR})\,\text{Perf}(D_{\text{clean}}, f_{\text{con}}) + \alpha\,\text{FNR}\,\text{Perf}(D_{\text{con}}, f_{\text{con}})}{(1-\alpha)(1-\text{FPR}) + \alpha\,\text{FNR}}, \tag{23}$$

which matches Eq. (9).

Equation 9 shows that **when contamination is non-trivial, detection-based filtering cannot eliminate inflation: any non-zero FNR leaves**

enough contaminated samples for inflation to persist as α increases. Our simulation (Figure 6a) and empirical results on an artificially contaminated MMLU model (Figure 6b) confirm this effect: even Min-K% with FNR@10%FPR $\approx$ 14% in our setting leaves substantial inflation as α increases, while other MIA baselines (e.g., loss/perplexity thresholding and likelihood-ratio methods (Carlini et al., 2022; Shi et al., 2023)) typically exhibit higher FNR@10%FPR in LLM settings.



(a) **Theoretical simulation.** (b) **Empirical measurement.**

Figure 6: **Left:** theoretical simulation of Eq. 9 at a fixed detector operating point. **Right:** empirical MMLU results showing the same trend when applying a Min-K% filter versus no filtering.

Why strong MIA is still insufficient in practice Our results with Min-K% indicate that even a strong LLM-oriented detector can leave substantial residual inflation under moderate-to-high contamination. This sensitivity is inherent to filtering: preventing inflation requires extremely small FNR at low FPR, which is difficult to achieve robustly across models, benchmarks, and prompting formats. Simpler detectors such as loss/perplexity thresholding and likelihood-ratio variants often yield higher FNR at comparable FPR in LLM settings, leading to even larger residual bias.

Selection bias and evaluation mismatch Detect-then-filter changes the evaluation distribution from D_{test} to D_{neg}. Even if the goal is to approximate clean evaluation, discarding items can bias the reported score because the retained subset is not guaranteed to be representative of the original benchmark, especially when FPR is non-negligible.

Implications for benchmark contamination While extreme leakage may be rare at the full-corpus level, widely reused benchmarks can accumulate exposure through repeated fine-tuning, dataset mixing, and web-crawled mirrors. Consequently, even moderate overall leakage can induce non-trivial effective contamination for popular subsets, where false negatives can dominate the residual inflation. This motivates inference-time interventions beyond filtering.

More simulation of MIA Figure 7 visualizes Equation (9) for different contamination rates α and false positive rates FPR. Even with a modest FNR on the x-axis, the measured accuracy on D_{neg} quickly departs from the clean baseline once α is moderate or large.

C Theorem 1

Theorem 1 (KL upper bound on performance gap)**.** *Fix a test set D_{test} and suppose*

$$\text{Perf}(f, D_{\text{test}}) = \mathbb{E}_{x \in D_{\text{test}}} \mathbb{E}_{y \sim p_f(\cdot|x)}[u(x, y)]$$

for some utility $u(x, y) \in [0, 1]$. Then, Δ_{perf} is bounded by:

$$\Delta_{\text{perf}} \leq \sqrt{\frac{1}{2}\mathbb{E}_x\Big[\text{KL}\big(p_{\text{con}}(\cdot|x) \,\|\, p_{\text{clean}}(\cdot|x)\big)\Big]}. \tag{24}$$

In particular, if the average KL divergence between $p_{\text{con}}(\cdot \mid x)$ and $p_{\text{clean}}(\cdot \mid x)$ is at most ε, then $\Delta_{\text{perf}} \leq \sqrt{\varepsilon/2}$.

Proof. For fixed x and any bounded $u(x, \cdot) \in [0, 1]$, standard arguments imply that the difference in expected utility is bounded by the total variation (TV) distance:

$$\begin{aligned}\big|\mathbb{E}_{p_{\text{con}}}[u(x, \cdot)] - \mathbb{E}_{p_{\text{clean}}}[u(x, \cdot)]\big| \\ \leq \text{TV}\big(p_{\text{con}}(\cdot \mid x), p_{\text{clean}}(\cdot \mid x)\big).\end{aligned} \tag{25}$$

Averaging over $x \in D_{\text{test}}$ yields an upper bound on Δ_{perf}. Applying Pinsker's inequality, $\text{TV}(P, Q) \leq \sqrt{\frac{1}{2}\,\text{KL}(P\|Q)}$, and then Jensen's inequality for the concave square-root function gives the stated bound in terms of the average KL divergence. □

D Generator Architecture Details

Overview Our perturbation generator consists of a network G_θ followed by a deterministic bounding layer. Given an input prompt x and its embedding sequence $e(x) \in \mathbb{R}^{L \times d}$ (computed by the contaminated model tokenizer/embedding layer), the generator first produces a raw perturbation

$$\tilde{\delta}(x) = G_\theta(e(x)) \in \mathbb{R}^{L \times d}, \tag{26}$$

and then applies a scaled $\tanh$ to obtain the bounded perturbation

$$\delta(x) = \zeta \cdot \tanh\big(\tilde{\delta}(x)\big), \tag{27}$$

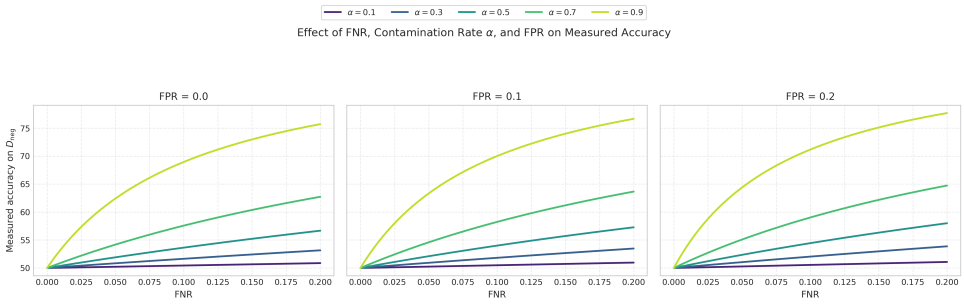


Figure 7: **Simulated effect of MIA errors.** Measured accuracy of the contaminated model f_{con} on D_{neg} as a function of FNR, for different contamination rates α and FPR values, computed from Equation (9) with $\text{Perf}(D_{\text{clean}}, f_{\text{con}}) = 50\%$ and $\text{Perf}(D_{\text{con}}, f_{\text{con}}) = 90\%$. Even small FNR values lead to substantial inflation when α is large.

which enforces $\|\delta(x)\|_\infty \le \zeta$ by construction. The perturbed embedding $e(x) + \delta(x)$ is then fed to f_{con} at inference time.

Unconstrained generator G_θ We instantiate G_θ as a lightweight decoder-only Transformer with n layers. Let $z = e(x) \in \mathbb{R}^{L\times d}$ be the input embedding sequence. We apply an input projection, n residual decoder blocks, and an output head:

$$\begin{aligned} H_0 &= zW_{\text{in}}, \\ H_\ell &= \text{Block}_\ell(H_{\ell-1}) \quad (\ell = 1, \dots, n), \\ \tilde{\delta}(x) &= H_n W_{\text{out}}, \end{aligned} \tag{28}$$

where $W_{\text{in}} \in \mathbb{R}^{d\times d_g}$ and $W_{\text{out}} \in \mathbb{R}^{d_g\times d}$ are learnable matrices and d_g is the hidden width of the generator (we use $d_g = d$ by default). The output $\tilde{\delta}(x)$ has the same shape as $e(x)$ to support token-wise perturbations.

Decoder block Each decoder block uses standard pre-norm Transformer components:

$$\begin{aligned} U_\ell &= H_{\ell-1} + \text{MHA}\big(\text{LN}(H_{\ell-1})\big), \\ H_\ell &= U_\ell + \text{FFN}\big(\text{LN}(U_\ell)\big), \end{aligned} \tag{29}$$

where MHA is masked multi-head self-attention, FFN is a position-wise feed-forward network, and LN is LayerNorm. We use the same causal attention mask as the base LLM so that the generator is compatible with autoregressive prompting.

Implementation choices Unless otherwise specified, we set $n = 4$ and use dropout 0.2 within the generator blocks. We found that this lightweight instantiation is sufficient to amortize per-instance perturbation optimization; our main results are not intended to depend on a sophisticated generator architecture, but rather on the reference-guided objective and the bounded embedding-space intervention.

Discussion The $\tanh$ bounding ensures a strict ℓ_∞ constraint and keeps the intervention local in embedding space. Alternative choices such as hard clipping yield similar behavior but introduce non-smooth gradients during training; we adopt the scaled $\tanh$ for stable optimization.

Algorithm 1: Training of the Noise Generator G_θ

Input: Auxiliary dataset D_{aux}; contaminated model f_{con}; reference model f_{ref}; learning rate $\eta > 0$; max norm $\zeta > 0$; Max iterations $T \in \mathbb{N}$; batch size $B \in \mathbb{N}$; optional stop threshold $\epsilon > 0$
Output: Trained parameters θ of G_θ

Initialize θ;
for $t = 1$ **to** T **do**
  Sample minibatch $\mathcal{B} = \{x_i\}_{i=1}^{B} \subset D_{\text{aux}}$;
  for $i = 1$ **to** B **do**
    $z_i \leftarrow e(x_i)$; // token embeddings
    $\delta_i \leftarrow \zeta \cdot \tanh\big(G_\theta(z_i)\big)$; // ℓ_∞-budgeted perturbation
    $p_i^{\text{ref}} \leftarrow f_{\text{ref}}(z_i)$; // reference output
    $p_i^{\text{con}} \leftarrow f_{\text{con}}(z_i + \delta_i)$; // perturbed contaminated output
  $\mathcal{L} \leftarrow \frac{1}{B}\sum_{i=1}^{B}\Big[\lambda_{\text{KL}}\,\text{KL}\big(p_i^{\text{con}} \,\|\, p_i^{\text{ref}}\big) + \lambda_{\text{CE}}\,\text{CE}\big(p_i^{\text{con}}, y_i^{\text{ref}}\big)\Big]$; // y_i^{ref} = hard labels from f_{ref}
  $\theta \leftarrow \theta - \eta\,\nabla_\theta \mathcal{L}$; // gradient descent
  if $\mathcal{L} \le \epsilon$ **then**
    break
return θ

E More Experiment

E.1 Definition of RC and BUD

We quantify the *decontamination effect* and the *benign side effect* of an inference-time mitigation operator $M(\cdot)$ using two complementary metrics. Let f_{con} denote the contaminated model under evaluation, and f_{clean} denote a reference model of the

Model	Split	Method	Access	TruthfulQA ($o = 1$)			MMLU ($o = 1$)		
				Exact	Semantic-level	Domain-level	Exact	Semantic-level	Domain-level
Mistral	f_{clean}	baseline	–	0.287	0.249	0.269	0.451	0.434	0.406
	f_{con}	W/O Decontamination	–	0.890 (0.603)	0.890 (0.641)	0.865 (0.596)	0.617 (0.166)	0.601 (0.167)	0.492 (0.086)
		TED	Black-box	0.785 (0.498)	0.770 (0.521)	0.736 (0.467)	0.404 (0.047)	**0.405 (0.029)**	**0.363 (0.043)**
		ITD	Black-box	0.890 (0.603)	0.890 (0.641)	0.865 (0.596)	0.610 (0.159)	0.588 (0.154)	0.499 (0.093)
		Shortcut Neuron	White-box	0.630 (0.343)	0.639 (0.390)	0.640 (0.371)	0.300 (0.151)	0.301 (0.133)	0.263 (0.143)
		Short Circuit	White-box	0.890 (0.603)	0.880 (0.631)	0.827 (0.558)	0.631 (0.180)	0.623 (0.189)	0.594 (0.188)
		DeconIEP(Ours)	White-box	**0.366 (0.079)**	**0.322 (0.073)**	**0.408 (0.139)**	**0.478 (0.027)**	0.467 (0.033)	0.470 (0.064)
Qwen 2.5	f_{clean}	baseline	–	0.555	0.531	0.572	0.670	0.671	0.673
	f_{con}	W/O Decontamination	–	0.866 (0.311)	0.856 (0.325)	0.784 (0.212)	0.775 (0.105)	0.769 (0.098)	0.735 (0.062)
		TED	Black-box	0.789 (0.234)	0.813 (0.282)	0.760 (0.188)	0.742 (0.072)	0.632 (0.039)	**0.619 (0.054)**
		ITD	Black-box	0.856 (0.301)	0.856 (0.325)	0.774 (0.202)	0.776 (0.106)	0.766 (0.095)	0.742 (0.069)
		Shortcut Neuron	White-box	0.306 (0.249)	0.297 (0.234)	0.274 (0.298)	0.395 (0.275)	0.417 (0.254)	0.389 (0.284)
		Short Circuit	White-box	**0.602 (0.047)**	**0.659 (0.128)**	**0.609 (0.037)**	**0.619 (0.051)**	**0.620 (0.051)**	0.610 (0.063)
		DeconIEP(Ours)	White-box	0.618 (0.063)	0.664 (0.133)	0.655 (0.083)	0.705 (0.035)	0.725 (0.054)	0.730 (0.057)
LLaMA-3	f_{clean}	baseline	–	0.474	0.467	0.452	0.576	0.565	0.549
	f_{con}	W/O Decontamination	–	0.900 (0.426)	0.909 (0.442)	0.885 (0.433)	0.734 (0.159)	0.728 (0.163)	0.663 (0.114)
		TED	Black-box	0.886 (0.412)	0.876 (0.409)	0.851 (0.399)	0.727 (0.151)	0.735 (0.170)	**0.576 (0.027)**
		ITD	Black-box	0.876 (0.402)	0.876 (0.409)	0.856 (0.404)	0.721 (0.145)	0.683 (0.118)	0.620 (0.070)
		Shortcut Neuron	White-box	0.804 (0.330)	0.789 (0.323)	0.644 (0.192)	**0.547 (0.029)**	**0.566 (0.001)**	0.484 (0.065)
		Short Circuit	White-box	0.589 (0.115)	0.612 (0.146)	0.635 (0.183)	0.695 (0.120)	0.682 (0.117)	0.710 (0.160)
		DeconIEP(Ours)	White-box	**0.517 (0.043)**	**0.476 (0.009)**	**0.461 (0.009)**	0.616 (0.040)	0.525 (0.041)	0.515 (0.034)

Table 5: Decontamination results for $o = 1$ across different models. Values denote accuracy, with parentheses showing Residual Contamination (RC) (smaller is better). Best RC is in bold and second best is underlined within each model and split.

same architecture. Let D_{con} be a contaminated (seen or near-seen) evaluation set, and D_{clean} be an uncontaminated evaluation set.

Formally,

$$\text{RC}(M) = \Big|\text{Perf}(f_{\text{clean}}, D_{\text{con}}) - \text{Perf}(M(f_{\text{con}}), D_{\text{con}})\Big|,$$

$$\text{BUD}(M) = \Big|\text{Perf}(f_{\text{con}}, D_{\text{uncon}}) - \text{Perf}(M(f_{\text{con}}), D_{\text{uncon}})\Big|.$$

E.2 The Full Result on Different Models under Different o

Evaluation splits (Exact / Semantic-level / Domain-level) For each benchmark, we report accuracy on three evaluation splits designed to separate memorization from generalization. **Exact** contains items that are exact matches (or maximally close duplicates) of leaked training instances, thus directly reflecting memorization-driven gains. **Semantic-level** contains meaning-preserving rewrites of the leaked items (e.g., paraphrases) that aim to break surface-form matching while retaining the same answer and comparable difficulty, thereby probing semantic memorization beyond verbatim recall. **Domain-level** consists of in-domain but non-leaked instances sampled to match the benchmark distribution, serving as an uncontaminated proxy for benign generalization.

How to read Tables 5 and 6 Rows are grouped by backbone model. Within each model, the f_{con} block reports the contaminated model under different inference-time mitigation methods, while the bottom row reports f_{clean} as a reference anchor. Each entry shows accuracy; the value in parentheses is **Residual Contamination (RC)**, computed as the absolute performance gap $\big|\text{Perf}(f_{\text{con}}) - \text{Perf}(f_{\text{clean}})\big|$ on the same split. Smaller RC indicates that the method more effectively removes contamination-induced inflation (i.e., the mitigated model behaves closer to the clean reference on contaminated splits).

Table 5 ($o = 1$): mild leakage regime When contamination severity is mild, W/O and rewriting-based baselines often retain elevated accuracy on **Exact** (and sometimes **Semantic-level**), indicating that memorization signals remain largely intact. DeconIEP consistently reduces **Exact** and **Semantic-level** accuracy toward the f_{clean} level, while keeping **Domain-level** accuracy comparatively stable. This pattern suggests a selective suppression of memorization-driven behavior without broadly impairing in-domain generalization. In contrast, activation patching and architectural

Model	Split	Method	Access	TruthfulQA ($o = 5$)			MMLU ($o = 5$)		
				Exact	Semantic-level	Domain-level	Exact	Semantic-level	Domain-level
Mistral	f_{clean}	baseline	–	0.287	0.249	0.269	0.451	0.434	0.406
	f_{con}	W/O Decontamination	–	0.995 (0.708)	1.000 (0.751)	0.885 (0.616)	0.995 (0.544)	0.972 (0.538)	0.463 (0.057)
		TED	Black-box	**0.086 (0.201)**	**0.244 (0.005)**	0.712 (0.443)	0.074 (0.377)	0.324 (0.110)	0.383 (0.023)
		ITD	Black-box	0.995 (0.708)	1.000 (0.751)	0.885 (0.616)	0.994 (0.543)	0.967 (0.533)	0.467 (0.061)
		Shortcut Neuron	White-box	0.621 (0.334)	0.601 (0.352)	0.640 (0.371)	0.394 (0.057)	0.409 (0.025)	0.245 (0.161)
		Short Circuit	White-box	0.919 (0.632)	0.914 (0.665)	0.736 (0.467)	0.581 (0.130)	0.535 (0.101)	**0.407 (0.001)**
		DeconIEP(Ours)	White-box	0.595 (0.308)	0.551 (0.302)	**0.570 (0.301)**	**0.454 (0.003)**	**0.423 (0.011)**	0.433 (0.027)
Qwen 2.5	f_{clean}	baseline	–	0.555	0.531	0.572	0.670	0.671	0.673
	f_{con}	W/O Decontamination	–	1.000 (0.445)	0.981 (0.450)	0.928 (0.356)	0.981 (0.311)	0.942 (0.271)	0.740 (0.067)
		TED	Black-box	0.191 (0.364)	0.632 (0.101)	0.788 (0.216)	0.870 (0.200)	0.837 (0.166)	0.886 (0.213)
		ITD	Black-box	1.000 (0.445)	0.981 (0.450)	0.928 (0.356)	0.983 (0.313)	0.938 (0.267)	0.752 (0.079)
		Shortcut Neuron	White-box	**0.517 (0.038)**	**0.507 (0.024)**	0.442 (0.130)	0.405 (0.265)	0.419 (0.252)	0.359 (0.314)
		Short Circuit	White-box	0.650 (0.095)	0.636 (0.105)	**0.677 (0.105)**	**0.750 (0.080)**	0.753 (0.082)	0.707 (0.034)
		DeconIEP(Ours)	White-box	0.790 (0.235)	0.799 (0.268)	0.718 (0.146)	0.805 (0.135)	**0.725 (0.054)**	**0.641 (0.032)**
LLaMA-3	f_{clean}	baseline	–	0.474	0.467	0.452	0.576	0.565	0.549
	f_{con}	W/O Decontamination	–	0.995 (0.522)	0.990 (0.524)	0.923 (0.471)	0.992 (0.416)	0.961 (0.395)	0.641 (0.092)
		TED	Black-box	0.866 (0.392)	0.866 (0.399)	0.870 (0.418)	0.794 (0.218)	0.785 (0.220)	0.610 (0.061)
		ITD	Black-box	0.909 (0.435)	0.928 (0.462)	0.870 (0.418)	0.824 (0.248)	0.828 (0.263)	0.610 (0.061)
		Shortcut Neuron	White-box	0.852 (0.378)	0.856 (0.390)	0.726 (0.274)	0.705 (0.130)	0.685 (0.120)	0.427 (0.122)
		Short Circuit	White-box	**0.589 (0.115)**	0.622 (0.155)	0.606 (0.154)	**0.697 (0.121)**	0.695 (0.130)	0.706 (0.157)
		DeconIEP(Ours)	White-box	0.660 (0.187)	**0.562 (0.096)**	**0.572 (0.120)**	0.736 (0.160)	**0.527 (0.038)**	**0.527 (0.022)**

Table 6: Decontamination results for $o = 5$ across different models. Values denote accuracy, with parentheses showing Residual Contamination (RC) (smaller is better). Best RC is in bold and second best is underlined within each model and split.

bypassing can reduce **Exact** as well, but their **Domain-level** degradation can be larger or less predictable, consistent with heavier interventions that overlap with general computation.

Table 6 ($o = 5$): strong memorization regime Under severe leakage, W/O frequently becomes near-saturated on **Exact** (and often remains high on **Semantic-level**), implying strong retrieval of leaked solutions. In this regime, the key question is whether a method can substantially reduce RC on contaminated splits without collapsing **Domain-level** performance. DeconIEP maintains a controlled reduction of contamination inflation (lower RC on **Exact/Semantic-level**) while avoiding catastrophic drops on **Domain-level**. Some baselines may become unstable or overly aggressive: stochastic calibration (TED) can severely hurt accuracy on certain settings, and broad architectural edits may trade stronger suppression for larger benign utility loss. "–" denotes missing runs for the corresponding method/model.

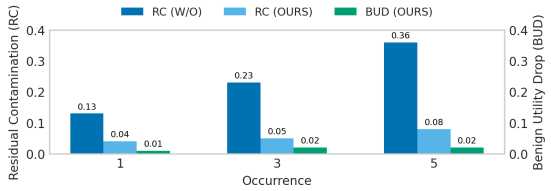
E.3 The result on Code Generation Task

Due to limited computational resources, we evaluate code-generation performance only on LLaMA-3-8b-instruct using HumanEval with 0-shot PASS@1. Figure 8 reports residual contamination (RC; lower is better) and benign utility drop (BUD; lower is better) as the contamination occurrence $o \in \{1, 3, 5\}$ increases.

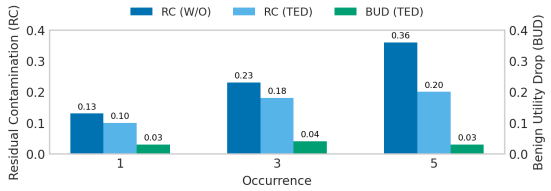
Without mitigation, RC increases monotonically with o (0.13→0.23→0.36), indicating stronger leakage effects under repeated exposure. Both TED and our method reduce RC across all o, but with markedly different trade-offs. TED yields only moderate RC reduction (0.10, 0.18, 0.20) and incurs non-trivial BUD (0.03–0.04). In contrast, our method substantially suppresses RC (0.04, 0.05, 0.08), achieving a larger absolute RC reduction (e.g., 0.36→0.08 at $o = 5$) while maintaining consistently smaller BUD (0.01–0.02). Overall, these results suggest that our inference-time perturbation better removes contamination-driven shortcuts while preserving benign code-generation utility under the same 0-shot Pass@1 evaluation.

E.4 RC–BUD Pareto trade-off

What Figure 9 shows Figure 9 visualizes the RC–BUD trade-off across methods. Each baseline appears as a *single point* because it does not expose a continuous knob analogous to our perturbation budget. DeconIEP forms a *curve* because varying ζ continuously changes the intervention strength.



(a) Ours



(b) TED

Figure 8: **HumanEval (0-shot Pass@1) under increasing contamination occurrence o.** We report residual contamination (RC; left axis, lower is better) and benign utility drop (BUD; right axis, lower is better) for LLaMA-3-8b-instruct. Compared to no mitigation, TED reduces RC with non-trivial BUD, while our method achieves larger RC reduction with smaller BUD across $o \in \{1, 3, 5\}$.

Interpreting the axes RC (horizontal or vertical, depending on plotting convention) measures how close the mitigated model aligns with the clean reference on contaminated evaluation, while BUD measures how much uncontaminated performance is lost due to the mitigation. Points closer to the origin are preferred. A method is Pareto superior if it achieves lower RC for the same (or lower) BUD.

Role of ζ Each point on the DeconIEP curve corresponds to a specific perturbation budget ζ. Smaller ζ yields conservative perturbations with low BUD but limited reduction in RC; larger ζ increases decontamination strength (lower RC) at the risk of higher BUD. This monotone, budget-controlled behavior enables users to select operating points that match application constraints (e.g., strict preservation of clean accuracy versus aggressive removal of contamination inflation).

Comparison to baselines By plotting baselines alongside the DeconIEP curve, the figure highlights whether DeconIEP can dominate existing approaches: if baseline points lie above and/or to the right of the curve, then for the same benign degradation DeconIEP achieves stronger decontamination, or for the same decontamination level it incurs less benign harm.

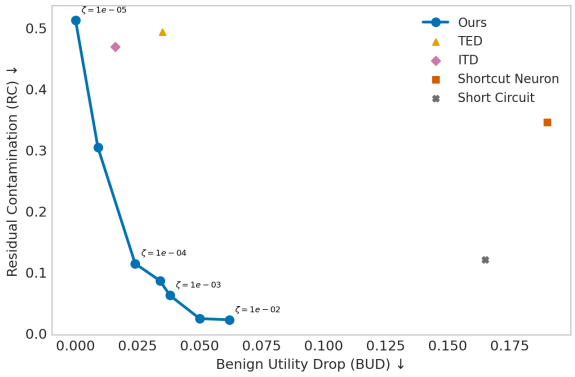
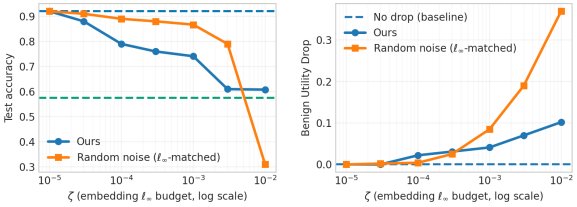


Figure 9: **RC–BUD Pareto trade-off induced by ζ.** Each point on the DeconIEP curve corresponds to a different perturbation budget ζ, illustrating controllable trade-offs between decontamination strength (RC) and benign utility drop (BUD). Baseline methods are shown as single points.



(a) Test accuracy vs. ζ . (b) Benign Utility Drop vs. ζ .

Figure 10: Learned perturbations vs. random noise across budgets ζ. (a) The blue and green dashed lines denote the unmitigated contaminated model and the clean-model baseline. (b) The blue dashed line denotes BUD$= 0$.

E.5 Learned perturbations vs. random noise

We test whether the gains come from structured, instance-adaptive perturbations rather than injecting noise with a comparable magnitude. We compare DeconIEP with an ablation that replaces the learned perturbation by *random noise* matched in ℓ_∞ norm and bounded by the same $\tanh$ operation. Figure 10 shows that random noise is substantially less reliable: while it can be competitive at very small budgets, its behavior becomes increasingly unstable as ζ grows, leading to inconsistent decontamination and larger BUD. In contrast, DeconIEP improves decontamination in a smooth and stable manner while incurring a smaller utility cost, indicating that effective decontamination requires targeted, instance-adaptive perturbations rather than unstructured noise.

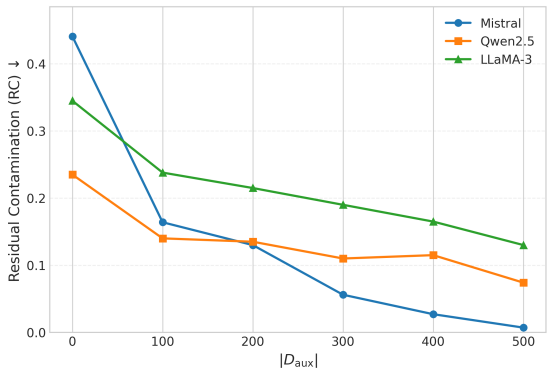


Figure 11: Ablation on the auxiliary set size $|D_{\text{aux}}|$ used to train the perturbation generator. We report Residual Contamination (RC; lower is better) as $|D_{\text{aux}}|$ increases for three backbones (Mistral, Qwen2.5, and LLaMA-3). Increasing $|D_{\text{aux}}|$ consistently reduces RC, indicating that additional auxiliary samples provide a stronger and more stable reference-guided training signal for decontamination.

E.6 Ablation on auxiliary set size

We study how the size of the auxiliary dataset $|D_{\text{aux}}|$ affects decontamination performance. $|D_{\text{aux}}|$ controls how many benchmark samples are used to train the perturbation generator under the reference-guided objective, and thus determines the strength and diversity of the supervision that steers the contaminated model away from memorization-driven behavior. As shown in Figure 11, RC decreases monotonically (or near-monotonically) as $|D_{\text{aux}}|$ increases across all three backbones, demonstrating that DeconIEP benefits from additional auxiliary samples and that the learned perturbations are not driven by a small set of idiosyncratic examples. The largest relative improvement typically occurs when moving from $|D_{\text{aux}}|{=}0$ to a small non-zero auxiliary set (e.g., 100), suggesting that even limited auxiliary data can provide sufficient signal to anchor the generator's behavior. Beyond this regime, further increasing $|D_{\text{aux}}|$ yields diminishing returns, consistent with the intuition that the generator begins to saturate once it has seen enough representative prompts to learn stable, instance-adaptive perturbation patterns. The trends differ by model family: Mistral shows the steepest early reduction in RC, while Qwen2.5 and LLaMA-3 improve more gradually, suggesting that these backbones may require more diverse auxiliary supervision to consistently suppress contamination-specific shortcuts. Overall, this ablation supports the practicality of our approach: strong decontamination can be achieved with a relatively small $|D_{\text{aux}}|$, while larger auxiliary sets further improve robustness without changing the discrete prompts or model parameters.

F AI Assistant Usage

We used AI assistants for language polishing and minor code scaffolding; all experimental design, implementation, and results were verified by the authors.