arXiv:2509.20909v2 [cs.CL] 9 May 2026

LogitTrace: Detecting Benchmark Contamination via Layerwise Logit Trajectories

Zirui He[1] **Haiyan Zhao**[1] **Yingcong Li**[1] **Ali Payani**[2] **Mengnan Du**[3*]
[1]New Jersey Institute of Technology
[2]Cisco Research
[3]The Chinese University of Hong Kong, Shenzhen
{zh296, hz54, yingcong.li}@njit.edu
apayani@cisco.com, mengnandu@cuhk.edu.cn

Abstract

Large language models (LLMs) are commonly evaluated on challenging benchmarks such as *AIME* and *Math500*, where benchmark contamination can make memorized solutions appear as genuine reasoning. Existing detection methods largely rely on surface overlap, completion behavior, or final-output likelihood, and often degrade when inputs are simply rephrased. In this paper, we propose **LogitTrace** (Layerwise Logit Trajectories), a framework for analyzing memorization-like decision dynamics through intermediate logit trajectories. Instead of judging memorization only from the final answer, LogitTrace examines how model preferences emerge and stabilize across layers. We find that contaminated examples tend to show earlier commitment, while clean examples exhibit more gradual evidence accumulation. These trajectory signals allow a lightweight classifier to separate contaminated and clean examples across multiple models and input variants. Controlled LoRA injection experiments further show that repeated exposure to target samples induces similar trajectory patterns. Overall, our results suggest that LogitTrace provides evidence beyond surface overlap and final-output confidence, offering a useful lens for studying memorization-like behavior in LLMs.

1 Introduction

Large language models (LLMs) have achieved remarkable performance across reasoning, dialogue, and code generation tasks [2, 19, 22]. However, their success has raised growing concerns about *data contamination*, where benchmark test data (or their transformed variants) appear in the training corpus. Such contamination can lead models to reproduce memorized solutions while appearing to reason, compromising claims of true generalization and raising privacy and intellectual property risks [3, 33, 17]. Contamination has been documented in widely used datasets such as MMLU and TruthfulQA [5]. It can yield deceptively strong outputs: models may still solve compressed or even invalid problems, reflecting recall rather than reasoning. Alternatively, Wu et al. [24] demonstrate that reinforcement learning gains on math benchmarks can vanish once contaminated items are removed, underscoring the urgent need for systematic contamination detection.

Existing detection methods primarily rely on completion-based recall tests [4, 17] and distributional probes such as perplexity and output agreement [15, 6, 21, 31, 30]. Other approaches include adversarial compression [20] and partial probing [33, 28]. While these methods provide valuable evidence of contamination, they often depend on surface forms, completion behavior, or final-output statistics. As a result, their robustness can degrade when test problems are rephrased, paraphrased, translated, or structurally altered [5, 27]. For example, Yang et al. [26] report that adding paraphrased

*Corresponding author.

Preprint.

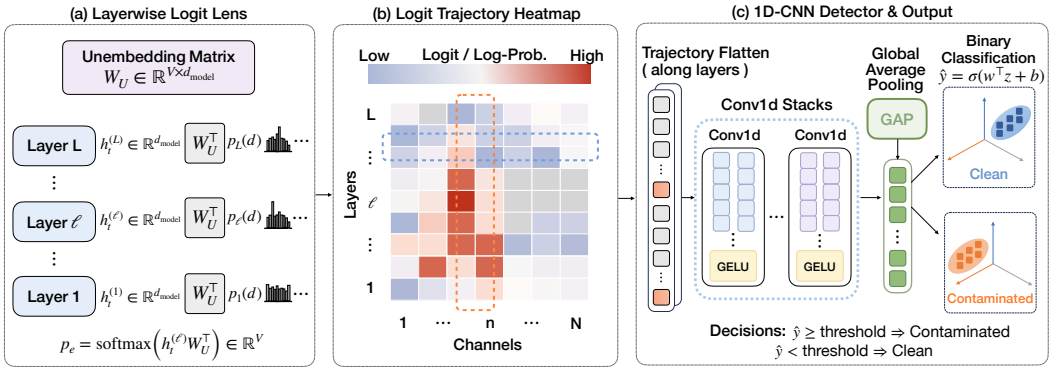


Figure 1: Overview of LogitTrace. (a) Intermediate hidden states are projected through the unembedding matrix to obtain layerwise digit probabilities. (b) For each sample, these trajectories form a layer-by-channel feature matrix, with rows denoting layers and columns denoting trajectory features. (c) The resulting matrix is fed as a multichannel sequence into a lightweight 1D-CNN detector, followed by global average pooling and a binary classifier for clean/contaminated prediction.

MMLU items into training raises Llama 2 accuracy from 45-55% to nearly 89%, almost matching GPT-4. Such findings reveal that contamination can persist even without verbatim overlap, motivating more robust representation-level detection methods.

To address these challenges, we introduce **LogitTrace** (Layerwise Logit Trajectories), a detection framework that identifies contamination-associated signals from layerwise logit dynamics. The key idea is that memorization may affect not only what the model answers, but also how the answer forms across layers. LogitTrace therefore tracks intermediate logit trajectories and tests whether their temporal structure separates contaminated from clean examples. We find a consistent pattern: contaminated examples tend to commit earlier, while clean examples accumulate evidence more gradually. A lightweight classifier built on these trajectories outperforms completion- and likelihood-based baselines under rephrasing, translation, and perturbation. Controlled LoRA injection experiments further show that repeated exposure to target samples strengthens similar trajectory signatures. Our contributions are threefold:

- We introduce LogitTrace, a detector that identifies contamination-associated decision dynamics from layerwise logit trajectories.
- We show that early commitment patterns distinguish contaminated from clean examples across multiple models and input variants.
- We use controlled LoRA injection to test whether repeated exposure induces the same type of trajectory signal observed in contaminated examples.

2 Preliminary

2.1 A Model-Level View of Memorization

Let Σ be the vocabulary and $\Sigma^\star$ the set of all finite sequences. A causal language model M_θ defines a conditional distribution $p_\theta(y \mid x)$ over continuations $y \in \Sigma^\star$ given a prompt $x \in \Sigma^\star$. We define the sequence-level log-likelihood as

$$s_\theta(y \mid x) \triangleq \sum_{t=1}^{|y|} \log p_\theta\big(y_t \,\big|\, x, y_{<t}\big). \tag{1}$$

Conceptually, a sequence $y^\star$ is memorized by M_θ if there exists a context x under which the model assigns $y^\star$ high likelihood and prefers it over plausible alternatives. Formally, for thresholds $\tau > 0$ and $\kappa \geq 0$,

$$\frac{1}{|y^\star|}\, s_\theta(y^\star \mid x) \geq \tau, \quad s_\theta(y^\star \mid x) - \max_{y \neq y^\star} s_\theta(y \mid x) \geq \kappa. \tag{2}$$

The first condition captures high likelihood, while the second imposes a margin against alternative continuations.

This view treats memorization as a model-level behavior rather than a directly observed dataset property. In modern LLMs, we usually cannot verify whether a benchmark item actually appeared in training data. We therefore use operational contamination labels and controlled injection experiments to study observable signatures associated with memorization-like behavior.

2.2 Explicit vs. Implicit Contamination

We study contamination at the data level, where benchmark items or their variants may appear in training data and affect evaluation. Following the model-level view above, such exposure can manifest during inference as memorization-like behavior. Since the exact training corpus is usually unavailable, we distinguish contamination regimes by the observable relationship between an evaluated benchmark item and a potentially exposed counterpart.

Explicit contamination. We refer to a benchmark item as *explicitly contaminated* when the evaluated instance preserves substantial surface overlap with a possibly exposed training instance. In this regime, detectors based on lexical similarity, completion behavior, or likelihood heuristics can often provide useful evidence [5, 17].

Implicit contamination. We refer to a benchmark item as *implicitly contaminated* when contamination effects persist even after the evaluated instance is transformed to reduce surface overlap, such as through rephrasing, translation, or structural perturbation. Recent findings show that surface-level signals are fragile in this setting: rephrased or translated benchmark items can still induce contamination effects even without verbatim overlap [26, 6]. This motivates detection strategies that capture subtler memorization dynamics beyond surface-level cues.

2.3 Representation Probing

Intermediate representations can reveal predictive information that is not visible from the final answer alone. The logit lens projects hidden states through the language modeling head to obtain layerwise vocabulary distributions [18], while the tuned lens improves this view with learned affine probes [1]. Related inversion and representation-decoding work further shows that next-token distributions and internal representations can preserve recoverable information about the input [16, 32]. Motivated by these findings, LogitTrace tracks answer-relevant digit probabilities across layers and tests whether contaminated samples exhibit distinct layerwise dynamics from clean samples.

3 Methodology

3.1 Problem Formulation

We consider an LLM M that takes a textual problem x as input and produces an answer a. Our objective is to detect whether an example (x, a) should be classified as *contaminated* or *clean* under an operational contamination setting. Formally, we construct a binary classification problem from a dataset $\{(x_i, a_i, y_i)\}_{i=1}^{N}$, where $y_i \in \{0, 1\}$ denotes the operational contamination label: $y_i = 1$ for contaminated examples and $y_i = 0$ for clean examples.

This formulation allows us to design a representation-based detector f_θ that takes intermediate activations from M as input and predicts a contamination score.

3.2 Representation Extraction

Building on the logit-lens view introduced in Section 2.3, we extract digit-level trajectories from the hidden states of M. Let $h_\ell \in \mathbb{R}^d$ denote the residual stream activation at layer ℓ. Applying the language modeling head W_{LM} yields logits and a predictive distribution over the vocabulary:

$$z_\ell = h_\ell W_{\text{LM}}, \quad p_\ell = \text{softmax}(z_\ell). \tag{3}$$

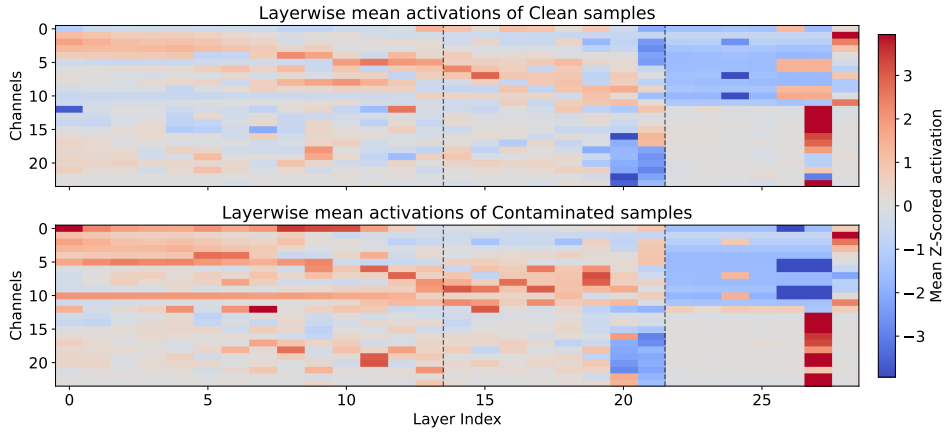


Figure 2: Class-mean activation heatmaps. Average channel activations across layers for clean vs. contaminated samples. Red regions denote stronger activation and blue denotes weaker activation.

Since we focus on numerical reasoning tasks, we restrict this distribution to digit tokens. Let $\mathcal{V}_d$ denote the set of token IDs corresponding to digit $d \in \{0, \ldots, 9\}$. The digit-only probability at layer ℓ is defined as:

$$p_\ell(d) = \frac{\sum_{t \in \mathcal{V}_d} p_\ell(t)}{\sum_{j=0}^{9} \sum_{t \in \mathcal{V}_j} p_\ell(t)}. \tag{4}$$

From this digit-level distribution, we compute two scalar statistics: digit entropy H_ℓ and maximum digit probability m_ℓ:

$$H_\ell = -\sum_{d=0}^{9} p_\ell(d) \log p_\ell(d), \quad m_\ell = \max_d p_\ell(d). \tag{5}$$

Together, these quantities describe how concentrated the model's digit preferences are at each layer, providing a compact view of layerwise commitment during answer formation.

3.3 Feature Construction

For each sample (x, a), we identify the first generation position corresponding to the answer and extract trajectories across the first T layers. At each layer ℓ, we concatenate digit probabilities, entropy, maximum digit probability, and their first-order differences:

$$X = \left[p_\ell(0..9),\ H_\ell,\ m_\ell,\ \Delta p_\ell,\ \Delta H_\ell,\ \Delta m_\ell\right]_{\ell=1}^{T}, \tag{6}$$

where Δ denotes first-order temporal differences, e.g., $\Delta H_\ell = H_\ell - H_{\ell-1}$. This results in a multichannel sequence representation $X \in \mathbb{R}^{C \times T}$, with $C = 24$ channels: 12 raw features and 12 difference features.

3.4 CNN-based Detector

We train a lightweight 1D convolutional neural network with hidden dimension $d = 128$ to classify whether a trajectory X is labeled as contaminated or clean. The network consists of three convolutional layers with ReLU activation and a global average pooling layer. The classifier produces logits

$$z = \text{CNN}(X) \in \mathbb{R}^d,\ o = Wz + b \in \mathbb{R}^2,\ \hat{y} = \text{softmax}(o). \tag{7}$$

The training objective is the standard cross-entropy loss, where $y_{i,c}$ denotes the one-hot operational label:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \sum_{c=1}^{2} y_{i,c} \log \hat{y}_{i,c}. \tag{8}$$

This detector tests whether the temporal structure of layerwise logit trajectories contains information that separates contaminated and clean examples beyond hand-crafted rules or single-layer statistics.

Table 1: Performance comparison of different methods across four text distributions. All numerical results are reported to one decimal place. Cont. means contaminated.

	Original		Rephrased		Translated		Perturbed	
	Cont. ↑	Clean ↓	Cont. ↑	Clean ↓	Cont. ↑	Clean ↓	Cont. ↑	Clean ↓
Qwen-2.5-7B								
Completion-based	100.0	11.6	16.7	3.7	12.2	2.6	15.8	3.0
TS-Guessing	92.7	14.5	21.3	15.0	3.6	0.0	17.7	14.6
Perplexity	87.6	1.1	29.2	0.4	10.8	0.7	19.0	0.0
Output Distribution	64.7	28.0	62.6	45.3	26.4	16.5	58.9	39.3
LogitTrace (Ours)	95.6	5.6	63.1	2.2	20.0	1.5	60.1	1.5
Qwen-2.5-Math-7B								
Completion-based	100.0	8.7	18.3	3.7	13.6	2.6	15.4	1.9
TS-Guessing	96.4	11.4	19.5	13.9	1.5	0.0	17.1	12.4
Perplexity	96.1	3.8	38.4	3.7	30.9	3.0	26.7	3.0
Output Distribution	71.3	19.3	50.5	20.9	17.8	11.2	50.5	26.6
LogitTrace (Ours)	89.4	5.6	56.2	2.6	64.2	9.4	51.7	1.9
Llama-3.1-8B								
Completion-based	100.0	14.4	4.3	6.3	3.2	3.4	2.9	4.8
TS-Guessing	98.2	17.3	29.1	0.3	0.3	0.2	0.3	0.2
Perplexity	93.7	11.2	36.1	10.6	31.5	8.7	24.9	8.1
Output Distribution	60.0	18.0	59.8	36.4	39.3	20.2	62.8	34.8
LogitTrace (Ours)	95.7	12.8	63.0	4.1	46.4	4.5	61.0	4.5
Qwen-3-8B								
Completion-based	100.0	14.2	19.4	2.6	13.8	2.6	18.8	1.9
TS-Guessing	91.3	21.7	20.1	19.5	3.8	1.9	19.0	17.6
Perplexity	70.0	39.7	63.7	42.3	84.4	64.8	57.5	36.7
Output Distribution	78.0	33.0	83.8	33.3	33.1	19.1	64.4	32.9
LogitTrace (Ours)	87.5	14.8	93.1	13.1	78.1	10.1	64.4	13.9

4 Experiments

In this section, we evaluate the performance of **LogitTrace** on memorization detection tasks to address the following research questions:

- **RQ1:** Do layerwise logit trajectories provide reliable signals for separating contaminated and clean examples, and are these signals robust to input variations?
- **RQ2:** Does controlled exposure through LoRA fine-tuning increase LogitTrace's contamination scores and induce similar trajectory patterns?
- **RQ3:** How do these contamination-associated signals depend on layer depth and detector architecture, and do they transfer to external benchmarks?

4.1 Experimental Setup

Models and Datasets. We evaluate four open-source LLMs including Qwen-2.5-7B, Qwen-2.5-Math-7B, Qwen-3-8B, and Llama-3.1-8B. Contaminated samples are constructed from *AIME 1983–2024* and *Math500* using a masked-completion procedure: the model is given partial problem text, and samples are retained when the generated completion exactly reconstructs the target text. Clean samples are constructed from *AIME 2025*, *Minerva Math*, and *LiveMathBench*, which were released after these models. To evaluate robustness, we additionally introduce three variants: *Rephrased* (paraphrased wording), *Perturbed* (surface-level edits), and *Translated* (French translation), by prompting GPT-4o-mini. We report the prompt details in Appendix B.

Baselines. We compare against four prior approaches. (i) *Completion-based* [4]: a sample is flagged as contaminated if the model's greedy completion exactly reconstructs the target text. (ii) *TS-Guessing* [5]: contamination is detected if the model can recover the target text from partial-information prompts or symbolic transformations. These two are completion-driven baselines. (iii) *Perplexity* [15]: computes the log-likelihood of the target text under the model; unusually low perplexity is taken as evidence of memorization. (iv) *Output Distribution* [6]: measures the similarity

Table 2: Detailed performance of LogitTrace across evaluation settings, reporting AUROC, PR-AUC, Precision, Recall, and F1.

Model	Setting	AUROC	PR-AUC	Precision	Recall	F1
Qwen-2.5-7B	Original	0.982	0.985	0.973	0.956	0.964
	Rephrased	0.944	0.972	0.983	0.631	0.769
	Translated	0.837	0.912	0.966	0.200	0.332
	Perturbed	0.948	0.974	0.988	0.601	0.748
Qwen-2.5-Math-7B	Original	0.962	0.979	0.975	0.894	0.933
	Rephrased	0.918	0.961	0.980	0.520	0.679
	Translated	0.900	0.953	0.944	0.642	0.764
	Perturbed	0.922	0.962	0.985	0.517	0.678
Llama-3.1-8B	Original	0.958	0.925	0.855	0.929	0.890
	Rephrased	0.927	0.890	0.902	0.630	0.742
	Translated	0.865	0.804	0.862	0.464	0.603
	Perturbed	0.925	0.885	0.899	0.610	0.727
Qwen-3-8B	Original	0.905	0.804	0.778	0.875	0.824
	Rephrased	0.960	0.929	0.810	0.931	0.866
	Translated	0.927	0.881	0.822	0.781	0.801
	Perturbed	0.949	0.910	0.796	0.900	0.845

between greedy decoding and multiple stochastic completions; higher similarity indicates stronger memorization-like behavior. These two are distribution-based baselines.

Metrics. We report detection metrics for the main contamination experiments. (i) *Youden's J index*: $J = \mathrm{TPR} - \mathrm{FPR}$, used to select the detector threshold by balancing contaminated detection and clean false positives. (ii) *Contamination Detection Rate*: the fraction of contaminated examples correctly flagged as contaminated at the chosen threshold. (iii) *False Positive Rate*: the fraction of clean examples incorrectly flagged as contaminated.

For the LoRA injection experiment, we additionally report reconstruction metrics to measure whether controlled exposure improves verbatim recovery of the target text. Specifically, we use a masked-completion setup in which the model observes a partial problem statement and is asked to complete the original text. (iv) *Exact Match (EM / pass@1)*: the fraction of first-generation completions that exactly match the target text. (v) *Rouge-L*: the longest common subsequence overlap between the generated completion and the target text, measuring partial reconstruction.

4.2 Main Results

Quantitative Analysis. We evaluate all methods on four text distributions: original, rephrased, translated, and perturbed. Table 1 shows clear differences across detection strategies. Completion-based approaches (Completion-based and TS-Guessing) perform well on original problems, but degrade sharply once the inputs are rephrased or perturbed, with contaminated detection rates often dropping below 20%. Distribution-based methods (Perplexity and Output Distribution) are somewhat less tied to exact completions, but remain unstable: Perplexity can produce high false positives on clean samples, while Output Distribution varies substantially across models and input variants. In contrast, LogitTrace generally provides a stronger trade-off between detecting contaminated examples and avoiding false positives on clean ones. For example, on rephrased Qwen-2.5-7B data, completion-based detection reaches only 16.7%, whereas LogitTrace reaches 63.1%; on perturbed Llama-3.1-8B data, Output Distribution mislabels over 30% of clean cases, while LogitTrace limits this to 4.5%.

Table 2 further reports threshold-independent and classification metrics for LogitTrace. Across most models and input variants, LogitTrace achieves high AUROC and PR-AUC, suggesting that the trajectory scores provide useful ranking signals beyond a single threshold. Precision is also consistently high, indicating that examples flagged as contaminated are rarely clean false positives.

Table 3: Detection and reconstruction results under LoRA fine-tuning with varying rank r. Contamination Detection Rate (CDR) denotes the fraction of target samples flagged as contaminated by the fixed detector. EM, Rouge-L, and Contamination Detection Rate are reported as percentages.

Model	Trainable-parameters	EM (pass@1)	RougeL	CDR
Pre-trained Model	0	17.8	35.4	2.2
Fine-tuned Model				
$r = 8$ (32 epochs)	20M	27.2	53.0	14.9
$r = 16$ (32 epochs)	40M	41.8	65.5	19.0
$r = 32$ (32 epochs)	80M	52.2	73.6	29.7
$r = 64$ (32 epochs)	170M	58.9	77.6	38.0
$r = 128$ (32 epochs)	340M	69.8	81.6	39.9
$r = 256$ (32 epochs)	670M	75.9	88.1	44.3
$r = 512$ (32 epochs)	1.3B	78.5	89.1	45.1

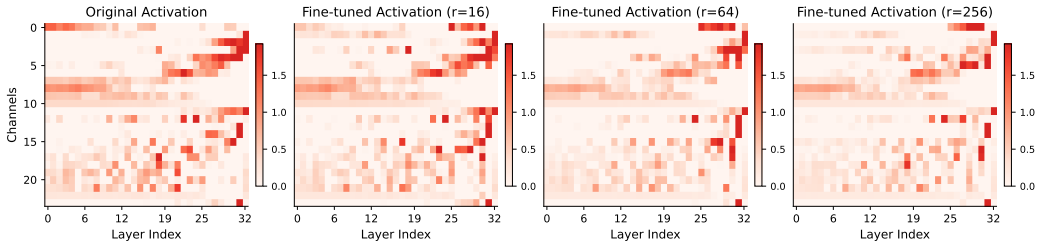


Figure 3: Layer-wise activations for one sample across LoRA ranks, with layers on the x-axis, 24 feature channels on the y-axis, and color showing normalized values.

The main degradation appears in recall and F1 under translated inputs for some models, especially Qwen-2.5-7B, suggesting that cross-lingual transformations remain more challenging than rephrasing or perturbation. Overall, these results indicate that layerwise logit trajectories provide robust contamination-associated signals.

Representation Analysis. We further examine class-mean activation heatmaps to understand how LogitTrace separates contaminated from clean examples. In Figure 2, the x-axis denotes model layers, the y-axis denotes the 24 trajectory feature channels, and color indicates the class-mean z-scored activation for each channel-layer pair. Compared with clean examples, contaminated examples show more pronounced high-activation regions in the early and middle layers. This pattern supports the layerwise-commitment hypothesis: contaminated examples tend to form answer-related preferences earlier in the forward pass.

4.3 LoRA Injection Evaluation

To test whether LogitTrace is sensitive to trajectory changes induced by controlled fine-tuning exposure, we conduct *LoRA injection experiments*. The detector f_θ is trained once and kept fixed throughout this experiment. We then fine-tune the base model M_{base} on the same target samples using LoRA with different ranks r, obtaining M_{lora}. A larger LoRA rank introduces more trainable adaptation parameters, allowing the fine-tuned model to fit the target samples more strongly. After fine-tuning, we apply the same fixed detector to the same samples under each LoRA model. If the detector score increases with r, this indicates that LogitTrace is sensitive to trajectory changes induced when the model has been trained on those samples.

As shown in Table 3, under M_{base}, these samples are almost always classified as clean, with a Contamination Detection Rate of only 2.2%. After LoRA fine-tuning, however, a larger fraction of samples are identified as contaminated, with the rate increasing to 44.3% at $r = 256$ and 45.1% at $r = 512$. This upward trend at small-to-moderate r is also aligned with reconstruction performance: both EM (pass@1) and Rouge-L steadily improve as r grows, suggesting that fine-tuning makes the model more capable of reconstructing the target text. Overall, the parallel increase in reconstruction

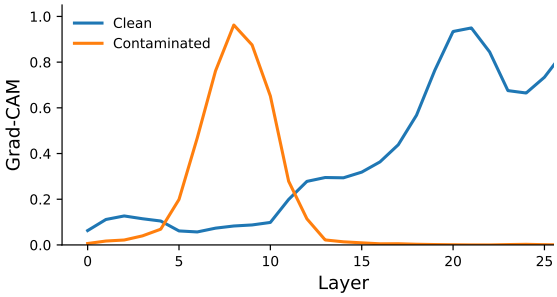


Figure 4: Class-mean Grad-CAM visualization comparing clean vs. contaminated samples.

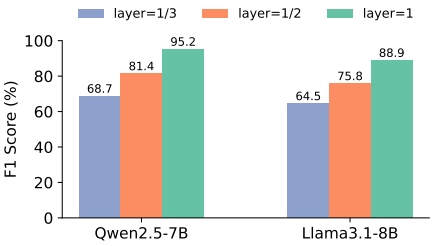


Figure 5: F1 scores across different layer depths.

performance and Contamination Detection Rate suggests that controlled exposure through LoRA fine-tuning strengthens the contamination-associated trajectory signals detected by LogitTrace.

In Figure 3, we present a controlled case study tracing a single evaluation sample across LoRA ranks. With the detector f_θ and threshold fixed, the predicted contamination probability increases systematically from 0.20 for the base model to 0.62 (r=16), 0.92 (r=64), and 0.98 (r=256). Since the detector, prompts, and test sample remain unchanged, this monotonic increase provides controlled causal evidence that LoRA exposure amplifies the trajectory signal used by the detector. The heatmaps illustrate how this signal emerges: in the base model, the feature channels remain relatively diffuse in the early layers without a clear preference. After LoRA fine-tuning, however, early-layer activations become more concentrated, and this concentration becomes more pronounced as r grows.

4.4 Ablation Study

Layer Selection. In the main experiments, detectors use activations from the full model depth. To assess where the relevant signal appears, we train additional detectors using only the first one-third or first half of the layers. As shown in Figure 5, the first one-third of layers already provides meaningful performance, with F1 scores of 68.7% on Qwen-2.5-7B and 64.5% on Llama-3.1-8B. Using the first half improves the scores to 81.4% and 75.8%, while using the full depth gives the best results, reaching 95.2% and 88.9%, respectively. These results suggest that contamination-associated signals emerge early, but aggregating evidence across depth yields the most reliable detector. Grad-CAM visualizations in Figure 4 support this trend: contaminated samples show sharper importance peaks in shallow layers, while clean samples have smoother importance distributions.

Detector Selection. To assess sensitivity to the detector architecture, we compare three lightweight detectors operating on the same multichannel sequence representation X under the same data split: our 1D-CNN detector, a two-layer MLP that averages features across depth, and a small two-layer Transformer encoder. As shown in Table 4, the 1D-CNN provides the best overall trade-off, achieving strong AUROC/PR-AUC (0.960/0.977) and the lowest False Positive Rate (5.56%). The MLP performs substantially worse, indicating that collapsing depth-wise trajectories discards critical information. The Transformer achieves comparable AUROC/PR-AUC and higher Contaminated Detection Rate, but incurs a substantially higher False Positive Rate (16.67%), suggesting a less favorable clean-side trade-off. Architectural and training details are deferred to Appendix D.

External Benchmark Generalization. Prior work has reported substantial contamination risks in MMLU, making it a useful external benchmark for testing whether LogitTrace captures contamination-associated structure beyond our constructed evaluation sets. We apply the detector trained on *Qwen-2.5-Math-7B* to the *mathematics subset of MMLU* in a zero-shot setting, without retraining or changing the detector threshold. Since item-level contamination labels are unavailable, we use the detector outputs for grouping: samples predicted as contaminated are referred to as *seen-like*, while those predicted as clean are referred to as *not-seen*.

As shown in Figure 6, LogitTrace labels 94.9% of the problems as *seen-like* and 5.1% as *not-seen*, which is consistent with prior reports that MMLU contains substantial contamination risk. More importantly, this split corresponds to a clear behavioral difference: *Qwen-2.5-Math-7B* achieves

Table 4: Detector architecture ablation using identical trajectory features. Cont. Detection Rate is the fraction of contaminated examples correctly flagged, while False Positive Rate is the fraction of clean examples incorrectly flagged.

Detector	Params (M)	AUROC	PR-AUC	Cont. Detection Rate (%)	False Positive Rate (%)
CNN	0.079	0.960	0.977	84.09	5.56
MLP	0.003	0.504	0.632	87.88	87.04
Transformer	0.069	0.954	0.974	95.45	16.67

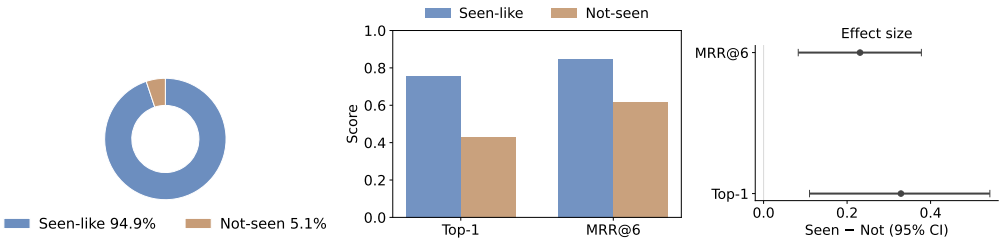


Figure 6: Dataset composition and group-level performance comparison. **Left:** fraction of samples classified as seen-like (94.9%) versus not-seen (5.1%). **Middle:** Top-1 accuracy and MRR@6 for the two groups. **Right:** effect sizes (Seen–Not) with 95% confidence intervals.

substantially higher performance on the *seen-like* group, with Top-1 accuracy higher by 0.33 and MRR@6 higher by 0.23. The confidence intervals indicate that these gaps are statistically reliable. These results suggest that LogitTrace transfers to an external benchmark and identifies a seen-like axis that correlates strongly with model performance, supporting its use as a tool for analyzing benchmark familiarity and contamination-associated behavior.

5 Related Work

Contamination. Benchmark results are often inflated when training and test overlap. Leakage has been documented in MMLU and other benchmarks [34, 5], while rephrased and translated variants can raise accuracy without verbatim overlap [26, 27]. Evaluation-side defenses attempt to mitigate these risks: CLEAN-EVAL and inference-time filtering sanitize benchmarks [36, 35], DCQ provides automated auditing [9], and task-level contamination undermines few-shot claims [14].

Detection and Mechanisms. Detection protocols formalize memorization as extractable or discoverable and extend to compression-based probing with shorter triggers [4, 17, 20], which capture verbatim recall yet still underestimate hidden forms under longer or rephrased inputs. Statistical calibration methods broaden coverage: minimum-probability detectors [21] and divergence-based scoring [31] flag anomalous likelihoods, paired confidence tests improve statistical reliability [30], and black-box calibration extends applicability to proprietary models [28]. Interventions can offer stronger causal evidence. For example, DICE operationalizes contamination by fine-tuning paired contaminated and uncontaminated models and then detecting in-distribution leakage from their internal representations [23] and temporal audits tracing when benchmarks leaked into training corpora [8]. Beyond detection, boundary studies interrogate whether high performance reflects reasoning or memorization, through idiom understanding [11], knowledge updating [13], generalization failures [6], and overestimation in machine translation [12]. Together, these studies suggest that, despite advances in coverage and robustness, existing methods remain largely tied to surface cues or distributional anomalies. This limitation motivates representation-level analysis that can more directly expose the underlying dynamics of memorization.

6 Conclusion

In this paper, we introduce LogitTrace, a representation-based framework for detecting benchmark contamination in LLMs. By analyzing layerwise logit trajectories, LogitTrace provides reliable

signals that remain stable under rephrasing and perturbation. Our comprehensive experiments, including controlled LoRA injection, not only confirm the robustness of this approach but also provide causal evidence that these trajectory patterns reflect model exposure rather than surface-level artifacts. Overall, LogitTrace offers a practical lens for studying memorization-like behavior and reassessing benchmark contamination in LLMs.

References

[1] Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens, 2025. URL https://arxiv.org/abs/2303.08112.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] 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. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[4] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

[5] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *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, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.482. URL https://aclanthology.org/2024.naacl-long.482/.

[6] Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. Generalization or memorization? data contamination and trustworthy evaluation for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12039–12050, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.716. URL https://aclanthology.org/2024.findings-acl.716/.

[7] André Vicente Duarte, Xuandong Zhao, Arlindo L. Oliveira, and Lei Li. De-cop: Detecting copyrighted content in language models training data, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/duarte24a.html.

[8] Shahriar Golchin and Mihai Surdeanu. Time travel in LLMs: Tracing data contamination in large language models. In *Proceedings of the 2024 International Conference on Learning Representations (ICLR)*, 2024.

[9] Shahriar Golchin and Mihai Surdeanu. Data contamination quiz: A tool to detect and estimate contamination in large language models. *Transactions of the Association for Computational Linguistics*, 13:809–830, 2025. doi: 10.1162/TACL.a.20.

[10] Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[11] Jisu Kim, Youngwoo Shin, Uiji Hwang, Jihun Choi, Richeng Xuan, and Taeuk Kim. Memorization or reasoning? exploring the idiom understanding of llms. In Christos Christodoulopoulos,

Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21678–21699, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1099. URL https://aclanthology.org/2025.emnlp-main.1099/.

[12] Muhammed Yusuf Kocyigit, Eleftheria Briakou, Daniel Deutsch, Jiaming Luo, Colin Cherry, and Markus Freitag. Overestimation in LLM evaluation: A controlled large-scale study on data contamination's impact on machine translation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=MpjtvkvXDo.

[13] Aochong Oliver Li and Tanya Goyal. Memorization vs. reasoning: Updating llms with new knowledge. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25853–25874, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1326. URL https://aclanthology.org/2025.findings-acl.1326/.

[14] Changmao Li and Jeffrey Flanigan. Task contamination: Language models may not be few-shot anymore. In *Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2024), Vancouver, Canada, February 20–27, 2024*, pages 18471–18480. AAAI Press, 2024. doi: 10.1609/AAAI.V38I16.29808. URL https://doi.org/10.1609/aaai.v38i16.29808.

[15] Yucheng Li. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation, 2023. URL https://arxiv.org/abs/2309.10677.

[16] John Xavier Morris, Wenting Zhao, Justin T Chiu, Vitaly Shmatikov, and Alexander M Rush. Language model inversion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=t9dWHpGkPj.

[17] Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from aligned, production language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vjel3nWP2a.

[18] Nostalgebraist. Interpreting GPT: The logit lens, 2020. Online blog post.

[19] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz

Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[20] Avi Schwarzschild, Zhili Feng, Pratyush Maini, Zachary Chase Lipton, and J Zico Kolter. Rethinking LLM memorization through the lens of adversarial compression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=KFmRMvzAZy.

[21] Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zWqr3MQuNs.

[22] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[23] Shangqing Tu, Kejian Zhu, Yushi Bai, Zijun Yao, Lei Hou, and Juanzi Li. Dice: Detecting in-distribution contamination in llm's fine-tuning phase for math reasoning, 2024. URL https://arxiv.org/abs/2406.04197.

[24] Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Huijie Lv, Ming Zhang, Yanwei Fu, Qin Liu, Songyang Zhang, and Qi Zhang. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination, 2025. URL https://arxiv.org/abs/2507.10532.

[25] Jialiang Xu, Shenglan Li, Zhaozhuo Xu, and Denghui Zhang. Do llms know to respect copyright notice? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[26] Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples, 2023. URL https://arxiv.org/abs/2311.04850.

[27] Feng Yao, Yufan Zhuang, Zihao Sun, Sunan Xu, Animesh Kumar, and Jingbo Shang. Data contamination can cross language barriers. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17864–17875, Miami, Florida, USA, November 2024. Association

for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.990. URL https://aclanthology.org/2024.emnlp-main.990/.

[28] Wentao Ye, Jiaqi Hu, Liyao Li, Haobo Wang, Gang Chen, and Junbo Zhao. Data contamination calibration for black-box llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10845–10861, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.644. URL https://aclanthology.org/2024.findings-acl.644/.

[29] Zhiyuan Yu, Yuhao Wu, Ning Zhang, Chenguang Wang, Yevgeniy Vorobeychik, and Chaowei Xiao. Codeipprompt: Intellectual property infringement assessment of code language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 40373–40389. PMLR, 2023.

[30] Huixuan Zhang, Yun Lin, and Xiaojun Wan. PaCoST: Paired confidence significance testing for benchmark contamination detection in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1794–1809, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.97. URL https://aclanthology.org/2024.findings-emnlp.97/.

[31] Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. Pre-training data detection for large language models: A divergence-based calibration method. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5263–5274, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.300. URL https://aclanthology.org/2024.emnlp-main.300/.

[32] Haiyan Zhao, Zirui He, Yiming Tang, Fan Yang, Ali Payani, Dianbo Liu, and Mengnan Du. Rep2text: Decoding full text from a single llm token representation, 2026. URL https://arxiv.org/abs/2511.06571.

[33] Weijie Zhao, Huajie Shao, Zhaozhuo Xu, Suzhen Duan, and Denghui Zhang. Measuring copyright risks of large language models via partial information probing, 2024. URL https://arxiv.org/abs/2409.13831.

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

[35] Qin Zhu, Qinyuan Cheng, Runyu Peng, Xiaonan Li, Ru Peng, Tengxiao Liu, Xipeng Qiu, and Xuanjing Huang. Inference-time decontamination: Reusing leaked benchmarks for large language model evaluation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9113–9129, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.532. URL https://aclanthology.org/2024.findings-emnlp.532/.

[36] Wenhong Zhu, Hongkun Hao, Zhiwei He, Yun-Ze Song, Jiao Yueyang, Yumeng Zhang, Hanxu Hu, Yiran Wei, Rui Wang, and Hongyuan Lu. Clean-eval: Clean evaluation on contaminated large language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 835–847, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.53. URL https://aclanthology.org/2024.findings-naacl.53/.

A Full CoT Outputs

These four examples illustrate the potential risk that memorization may masquerade as generalization. In Table 5, the model answers the original problem correctly, which is expected. In Table 6, after rephrasing, the model still provides the correct solution, appearing to generalize. In Table 7, however, even when the input is reduced to a minimal set of keywords, the model produces the correct answer together with a full chain-of-thought explanation. This behavior is unlikely to arise from genuine reasoning and instead suggests direct recall of memorized content. In Table 8, the model attempts to generate an answer even for an invalid input, further underscoring its reliance on pattern matching rather than reasoning. These cases reveal that model outputs may reflect hidden memorization signals disguised as generalization, which poses potential risks in downstream use.

Table 5: Case study 1: Original baseline question.

Case Study 1: Original Question (Baseline)
Question: A regular pentagon is rotated counterclockwise about its center. What is the minimum number of degrees it must be rotated until it coincides with its original position?
Model Output: 72 degrees. To determine the minimum number of degrees a regular pentagon must be rotated counterclockwise about its center until it coincides with its original position, we need to consider the symmetry of the pentagon. A regular pentagon has rotational symmetry, meaning it looks the same after being rotated by certain angles around its center. The order of rotational symmetry for a regular pentagon is 5. This means there are 5 different positions where the pentagon looks exactly the same when rotated. The angle of rotation for each symmetry position is given by dividing 360 degrees (a full rotation) by the order of symmetry. So, we calculate: $$\frac{360^\circ}{5} = 72^\circ$$ Therefore, the minimum number of degrees the pentagon must be rotated counterclockwise until it coincides with its original position is $\boxed{72}$.

Table 6: Case study 2: Hexagon input, model outputs correct result.

Case Study 2: Core Number Changed (Hexagon)

Question:
A regular hexagon is rotated counterclockwise about its center. What is the minimum number of degrees it must be rotated until it coincides with its original position?

Model Output:
60 degrees.
To determine the minimum number of degrees a regular hexagon must be rotated counterclockwise about its center until it coincides with its original position, we need to consider the symmetry of the hexagon. A regular hexagon has rotational symmetry, meaning it looks the same after being rotated by certain angles around its center.
A regular hexagon has 6 equal sides and 6 equal angles. The sum of the interior angles of a hexagon is given by the formula $(n-2) \times 180^\circ$, where n is the number of sides. For a hexagon ($n = 6$), the sum of the interior angles is:

$$(6-2) \times 180^\circ = 4 \times 180^\circ = 720^\circ$$

However, we are interested in the angles of rotation that will map the hexagon onto itself. Since the hexagon has 6 equal sides, it can be divided into 6 equal parts. Therefore, the angle of rotation that will map the hexagon onto itself is:

$$\frac{360^\circ}{6} = 60^\circ$$

This means that if the hexagon is rotated by 60 degrees counterclockwise, it will coincide with its original position. any multiple of 60 degrees (i.e., 120 degrees, 180 degrees, 240 degrees, 300 degrees, 360 degrees, etc.) will also map the hexagon onto itself, but the minimum positive angle is 60 degrees.
Thus, the minimum number of degrees the hexagon must be rotated until it coincides with its original position is $\boxed{60}$.

Table 7: Case study 3: Keyword-based input, model still outputs the correct reasoning.

Case Study 3: Keyword Summary Input

Question:
minimum rotation degrees, regular pentagon, symmetry

Model Output:
72°
The minimum rotation degrees that will map a regular pentagon onto itself is 72°. This is because a regular pentagon has five sides, and a full rotation of 360° divided by five gives 72°. This means that rotating the pentagon by 72° will bring it back to a position where it looks exactly the same as it did before the rotation.
To understand this better, let's consider the symmetry of a regular pentagon. A regular pentagon has five lines of symmetry, each passing through a vertex and the midpoint of the opposite side. When the pentagon is rotated by 72°, each vertex moves to the position of the next vertex in a clockwise or counterclockwise direction. This rotation aligns the pentagon with itself, leaving it unchanged.
Therefore, the minimum rotation degrees that will map a regular pentagon onto itself is $\boxed{72^\circ}$.

Table 8: Case study 4: Input with non-integer polygon sides.

Case Study 4: Input Floating Number (4.5)

Question:
A regular 4.5-sided polygon is rotated counterclockwise about its center. What is the minimum number of degrees it must be rotated until it coincides with its original position?

Model Output:
80 degrees.
To determine the minimum number of degrees a regular 4.5-sided polygon must be rotated until it coincides with its original position, we start by understanding the properties of regular polygons and their rotational symmetry.
A regular polygon with n sides can be rotated by $\frac{360^\circ}{n}$ degrees to coincide with its original position. This is because a full rotation of 360° degrees will bring the polygon back to its starting position, and dividing this by the number of sides gives the smallest angle of rotation that will result in the polygon coinciding with itself.
In this problem, the polygon has 4.5 sides. However, a polygon must have an integer number of sides. Therefore, we need to interpret the problem as referring to a polygon with a fractional number of sides in a theoretical or conceptual sense, and we can use the same formula to find the angle of rotation.
The formula for the angle of rotation is:

$$\text{Angle of rotation} = \frac{360^\circ}{4.5}$$

To simplify this, we can convert 4.5 to a fraction:

$$4.5 = \frac{9}{2}$$

Now, substitute this into the formula:

$$\text{Angle of rotation} = \frac{360^\circ}{\frac{9}{2}} = 360^\circ \times \frac{2}{9} = \frac{360^\circ \times 2}{9} = \frac{720^\circ}{9} = 80^\circ$$

Thus, the minimum number of degrees the polygon must be rotated until it coincides with its original position is:

$$\boxed{80}$$

B Rephrase dataset prompt

B.1 Problem Transformation Setup

We applied controlled prompting to generate different variants of math problems for our experiments. All generations were obtained using the `GPT-4o-mini` model (via the `chat.completions` API). To ensure variability while preserving mathematical correctness, we set the sampling temperature to 0.7.

We designed three transformation tasks:

- **Rephrasing**: synonym replacement or structure change without altering semantics.
- **Perturbation**: minor linguistic noise while keeping numbers/LaTeX fixed.
- **Translation (French)**: translate only the surrounding text, keeping numbers/LaTeX unchanged.

The exact prompts used for each task are shown in Table 9.

Table 9: Prompt templates used to generate rephrased, perturbed, and translated math problems. All generations used GPT-4o-mini with temperature $= 0.7$.

Task	**Prompt Template**
Rephrase	Rephrase the following math problem in English. Requirements: 1. Keep ALL numbers and LaTeX expressions exactly the same. 2. Do not change the mathematical meaning. 3. Output ONLY the rephrased problem, no explanations. Problem: {q}
Perturb	Rewrite the following math problem with minor linguistic perturbations. You may change word order, replace synonyms, or alter sentence structure. Requirements: 1. Keep ALL numbers and LaTeX expressions exactly unchanged. 2. Preserve the exact mathematical meaning. 3. Output ONLY the perturbed problem. Problem: {q}
Translate (French)	Translate the following math problem into French. Requirements: 1. Keep ALL numbers and LaTeX expressions unchanged. 2. Translate only the surrounding text. 3. Output ONLY the translated problem, no explanations. Problem: {q}

C More Activation Trajectories Across Models

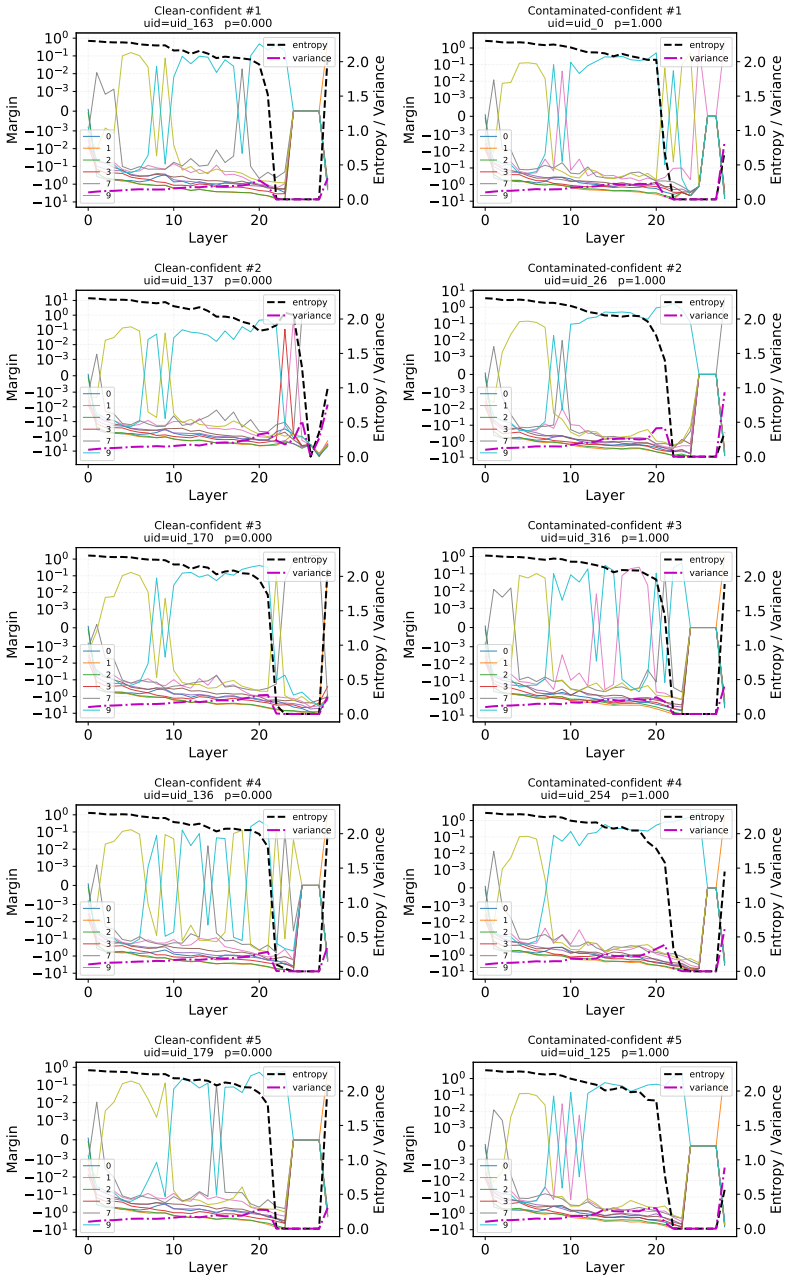


Figure 7: Activation trajectories of the 10 most confidently classified samples by the detector trained on Qwen-2.5-7B. Clean samples are shown on the left, while contaminated samples are shown on the right.

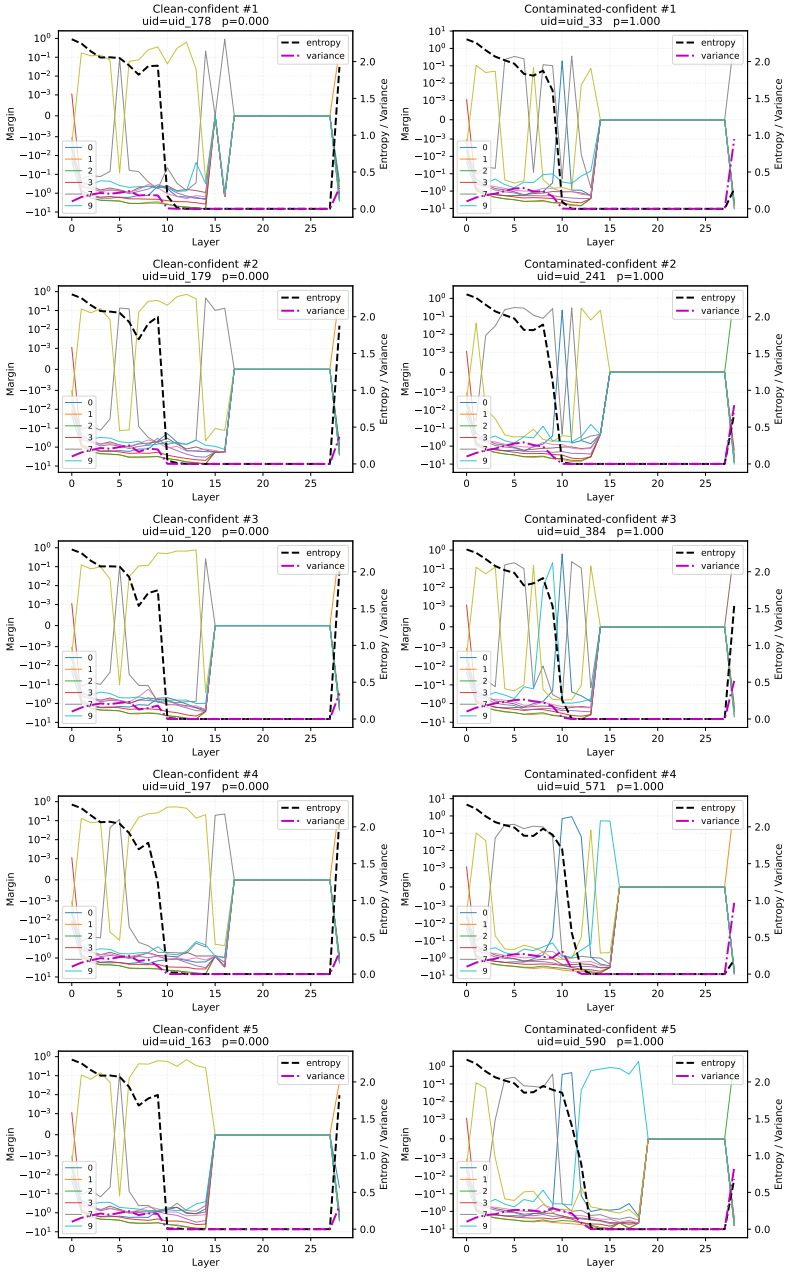


Figure 8: Activation trajectories of the 10 most confidently classified samples by the detector trained on Qwen-2.5-Math-7B. Clean samples are shown on the left, while contaminated samples are shown on the right.

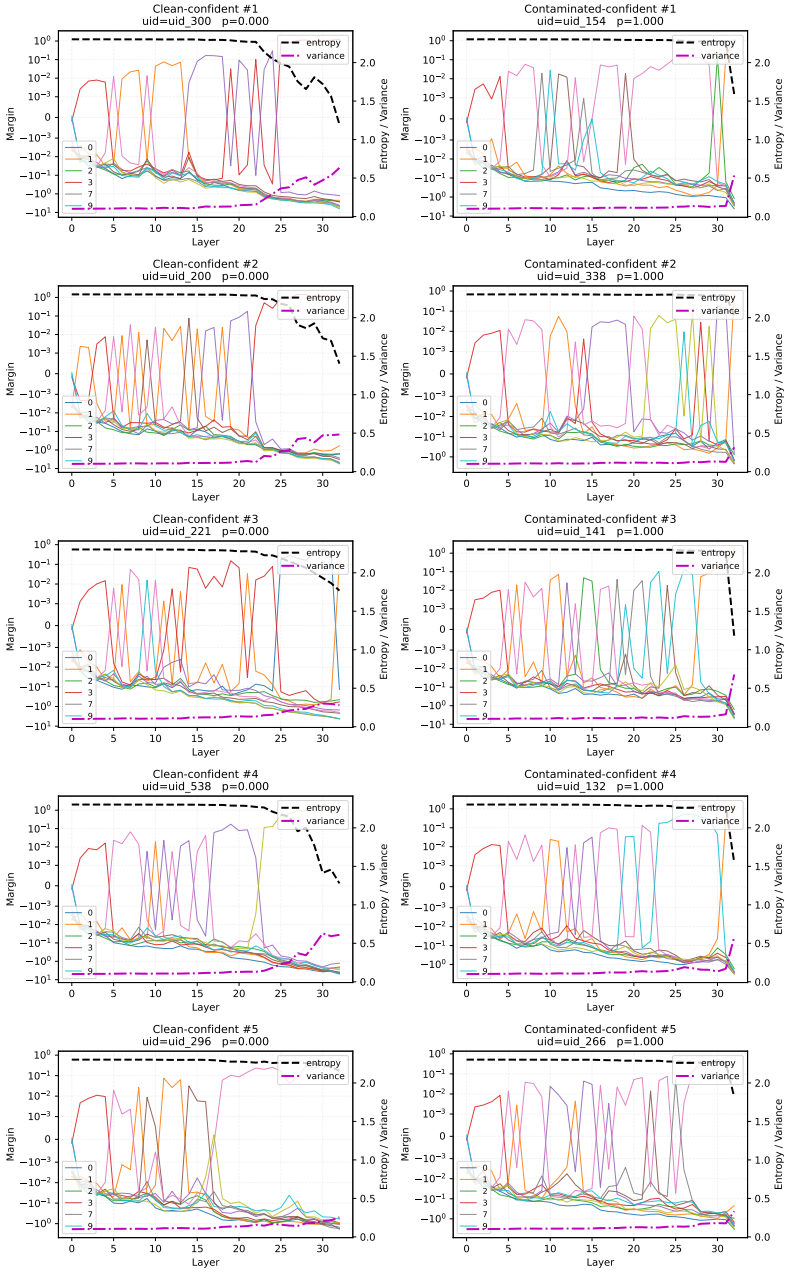


Figure 9: Activation trajectories of the 10 most confidently classified samples by the detector trained on Llama-3.1-8B. Clean samples are shown on the left, while contaminated samples are shown on the right.

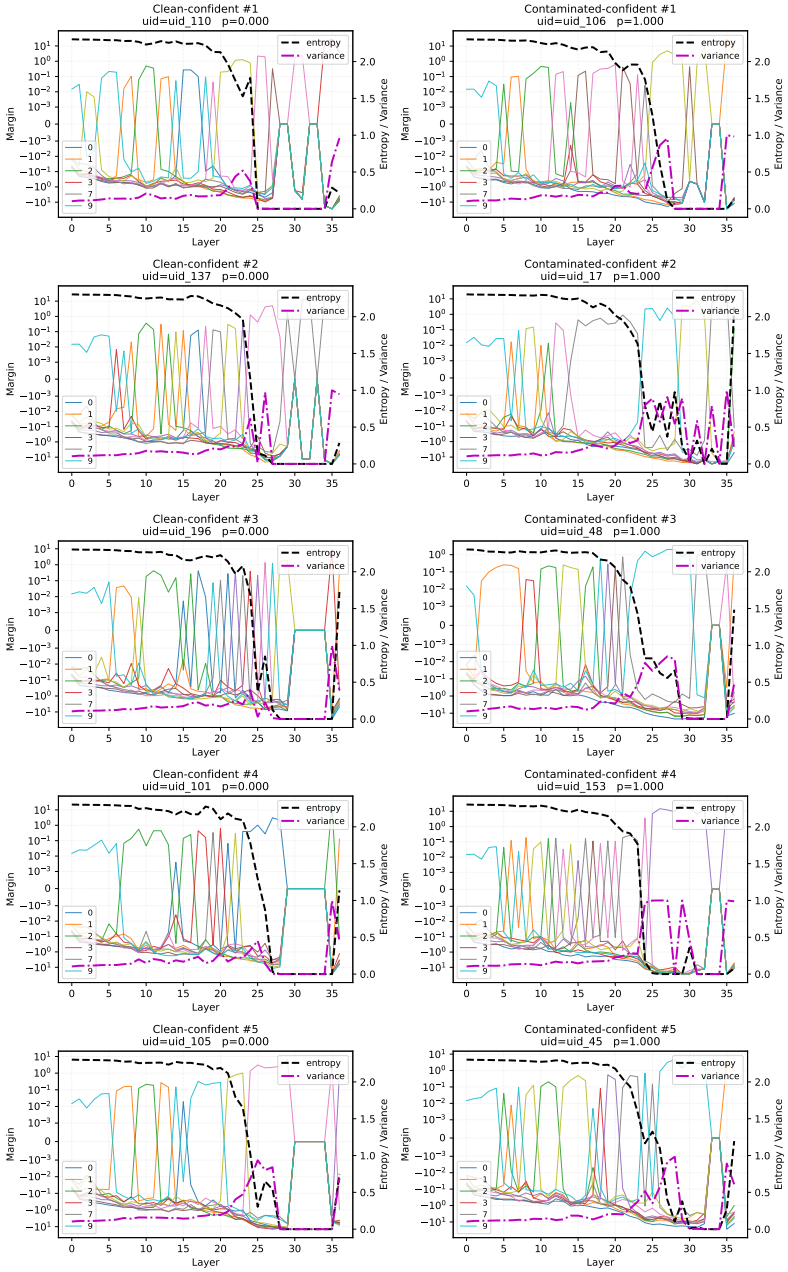


Figure 10: Activation trajectories of the 10 most confidently classified samples by the detector trained on Qwen-3-8B. Clean samples are shown on the left, while contaminated samples are shown on the right.

D Implementation Details

D.1 Detector Training

Feature construction. For each problem–answer pair, we tokenize the prompt ‘‘`Problem: <q>\nAnswer:`’’ with the model's native tokenizer and run the model with `output_hidden_states=True`. At every layer l, we compute (i) the normalized probability mass over digit tokens $p_l(d)$ for $d \in \{0, \ldots, 9\}$ (token sets collected from the tokenizer vocabulary), (ii) the entropy H_l of p_l, and (iii) the maximum probability $\max_d p_l(d)$. We also take first-order differences along depth for each of these streams. Stacking gives a $C \times T$ trajectory with $C = 24$ channels: 10 digit channels $p_l(d)$, plus H_l and $\max p_l$, and their 12 first-order differences. We z-score normalize *per sample, per channel* (NaNs $\to$ 0). For sequences with different T, we truncate all samples to the minimal T across the batch to align time axes.

Detector architecture. We use a lightweight 1D CNN (`TinyTSConv`) with three `Conv1d` layers (kernel size 3, channels $C \to 64 \to 128 \to 128$) and ReLU activations, followed by global average pooling and a 2-way linear head. Inputs have shape $[N, C, T]$ with $C = 24$.

Training protocol. We perform a stratified split with validation ratio 0.2. Optimization uses AdamW with learning rate 2×10^{-3}, batch size 32, and cross-entropy loss. To address class imbalance, we apply inverse-frequency class weights $w_k = \dfrac{n_{\text{pos}} + n_{\text{neg}}}{2 \max(n_k, 1)}$. We train for 15 epochs and select the checkpoint that maximizes AUROC on the validation set. All random seeds are fixed to 42 (Python/NumPy/PyTorch). Training runs on the same device as the backbone model (CPU/GPU), with features stored as `float32`.

Detector ablation. Regarding the choice of a CNN-based detector, we conducted an additional ablation study comparing the CNN with a two-layer MLP and a lightweight 2-layer Transformer encoder ($d_{\text{model}} = 64$, $n_{\text{head}} = 4$), all trained on the same representation-trajectory features and under the same train/validation split.

The MLP aggregates the $[C, T]$ feature map by averaging over the temporal dimension to produce a C-dimensional vector, followed by two linear layers with a ReLU activation in between.

The Transformer detector begins by projecting each layer-wise feature vector from C to a 64-dimensional embedding. This sequence of T embeddings is then fed into a small encoder consisting of two 4-head self-attention layers with a 128-dimensional feed-forward block and the standard GELU activation used in PyTorch's implementation. After encoding, the model averages over the temporal dimension and applies a final linear classifier. Both alternatives have comparable parameter counts to the CNN (approximately 0.003M for the MLP and 0.07M for the Transformer versus 0.079M for the CNN).

D.2 Lora Fine-tuning Experiment

LoRA fine-tuning. We fine-tune *Llama-3.1-8B* using LoRA adapters applied to attention and MLP projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`). Training is performed in 4-bit NF4 quantization with double quantization, and all experiments vary the LoRA rank r while setting $\alpha = 2r$ and dropout 0.05. Problems are concatenated and segmented into blocks of 1024 tokens, where the model is trained in the standard causal language modeling objective with cross-entropy loss. We use AdamW with learning rate 1×10^{-4}, cosine decay, warmup ratio 0.03, batch size 2 per device with gradient accumulation 8, and train for 32 epochs. Gradient checkpointing and TF32 are enabled. All random seeds are fixed to 0 for reproducibility.

E More Related Work

Copyright. A practical data contamination scenario is copyright. DE-COP identifies copyrighted content in pretraining corpora [7], audits reveal explicit copyright violations in model outputs [10], partial probing quantifies exposure risk [33], and further tests examine whether models comply with copyright notices or reproduce licensed code [25, 29]. Such findings underscore that contamination

is not only a technical artifact but also a legal and ethical risk: reproducing protected material can mislead benchmark evaluations and at the same time threaten compliance with intellectual property.

F Limitations

LogitTrace uses operational contamination labels because the true pretraining corpus of each model is unavailable, LogitTrace relies on operational contamination labels. Still, the consistency of its trajectory patterns across models, input variants, and controlled exposure experiments suggests that the detector captures meaningful contamination-associated signals. Its predictions should therefore be interpreted as diagnostic evidence rather than proof that a sample appeared in training data. The current experiments focus on numerical reasoning and digit-level trajectories, so extensions to open-ended or non-numeric tasks may require new task-specific token sets and feature definitions. In addition, LogitTrace requires access to intermediate model representations and layerwise logits, which limits its direct applicability to closed-source models that expose only final outputs.