FIGHTING FIRE WITH FIRE: ASSESSING TEST SET CONTAMINATION VIA DELIBERATE TRAINING ON TEST DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Test set contamination inflates benchmark scores and misrepresents what a model can do. Existing detectors read a trained model's static outputs, which a developer can optimize against directly. We audit by intervention instead: fine-tune the suspect model on the benchmark in question and read the shape of its loss curve. A model re-exposed to data it has seen before descends differently than one meeting that data for the first time, and the difference survives zero-normalizing every curve, so it lives in the shape of learning rather than in the loss level. We introduce FIRE (Fine-tuning-Induced Re-Exposure), which classifies these curves with dynamic time warping and k-nearest neighbors: no learned parameters, no model internals, one fine-tuning run (about ten minutes for a 7B model on a single GPU) and a lookup against a reference library built once per benchmark. Across six models (1B–7B), 44 benchmarks, and over 43,000 training runs, FIRE reaches 77–92% accuracy at contamination levels that meaningfully inflate benchmark scores, matching or beating five membership inference baselines and beating every baseline that does not require a clean same-architecture model. The signal transfers across model families at 87%, reaches a 70B model at 69–78% zero-shot and 88.7% with 50 labeled trajectories from it, and persists through a subsequent preference-optimization (DPO) stage with clean data. Detection needs as few as 256 benchmark samples, while evading it requires a full probing run inside every poisoning step. What a model has already seen is legible in how it learns.

1 INTRODUCTION

Test set contamination fundamentally undermines model evaluation by violating the core assumption that evaluation data remains unseen during training. When evaluation benchmarks leak into training corpora—whether through inadvertent web scraping or deliberate inclusion—the resulting performance metrics cease to reflect true model capabilities. This contamination problem has intensified as large language models (LLMs) train on increasingly vast web-scraped datasets that may contain evaluation benchmarks, producing inflated performance reports that mislead researchers and practitioners deploying these models in real-world applications.

Existing contamination detection methods face critical limitations. Output distribution analysis (Yang et al., 2023), perplexity-based metrics (Dong et al., 2024), and behavioral probes such as the Data Contamination Quiz (Golchin & Surdeanu, 2025) and "Time Travel" guided-instruction prompts (Golchin & Surdeanu, 2024) analyze models in their final trained state. A developer who knows the detection criterion can therefore target it directly, suppressing the measured signal while keeping the benefit of contamination. Membership inference attacks (Shokri et al., 2017; Yeom et al., 2018; Carlini et al., 2022; Shi et al., 2024) address a related but distinct problem: whether a specific data point was in the training set. However, contamination detection requires a dataset-level judgment of whether a benchmark as a whole influenced training, which is not reducible to aggregating per-sample membership decisions, particularly when contamination does not produce strong item-level memorization.

A model re-exposed to data it has seen before learns it differently than a model meeting that data for the first time. Our approach, FIRE (Fine-tuning-Induced Re-Exposure), is straightforward: fine-

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

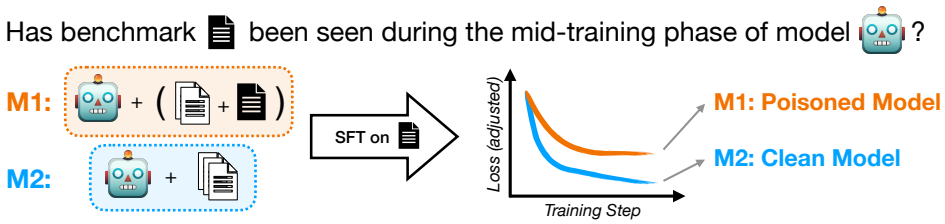


Figure 1: Overview of our auditing procedure. Given a suspect model, we fine-tune it on the target benchmark $\mathcal{B}$ and record the loss trajectory. A classifier operating solely on the trajectory's shape determines whether $\mathcal{B}$ (or data drawn from the same distribution) was seen during the model's original training.

tune the suspect model on the benchmark in question and record the per-step training loss. The resulting trajectories are measurably distinct. Specifically, we retain only contamination that materially inflates benchmark scores, and classify the resulting loss curves using a simple non-parametric classifier operating solely on trajectory shape. Crucially, the reference library transfers across held-out benchmarks and model architectures, and reliable detection requires as few as 256 benchmark samples for probing.

We validate this approach at scale: six models, 44 benchmarks, and over 43,000 training runs. FIRE uses only loss curves from a short fine-tuning run, without access to the model's original training data or recipe, yet achieves 77–92% detection accuracy at contamination levels that meaningfully inflate benchmark scores ($\Delta \geq 0.05$), matching or outperforming all five MIA baselines we evaluate, and beating every baseline that does not require a clean model of the same architecture. It is interpretable by design: k-medoids clustering produces concrete exemplars of clean versus contaminated trajectories, making classification decisions auditable. The signal generalizes across model families at 87% accuracy, requires as few as 256 evaluation samples, is detectable with in-distribution data even when the exact leakage is not known, and persists even when mid-training contamination is followed by post-training via DPO (84% detection). In a separate scaling study, we show that classifiers trained on the six 1B–7B models transfer zero-shot cross-scale to a 70B model at up to 77.7% accuracy and reach 88.7% after adding only 50 labeled trajectories.

These results establish training dynamics as a robust auditing signal. Because the signal emerges from the interaction between model weights and gradient updates during fine-tuning, optimizing against it is expensive: the adversary must solve a bi-level problem in which every poisoning update requires a full probing run to obtain the loss curve, an $\sim 850\times$ cost multiplier over the poisoning run itself (§4.5). Auditing, by contrast, costs one fine-tuning run and a single classifier lookup, and any criterion derived from a single forward pass yields evasion gradients for free. This distinction is important as models train on increasingly vast web-scraped corpora where benchmark leakage is difficult to prevent and static approaches, including exchangeability tests (Oren et al., 2024), may be vulnerable to adaptive evasion. More broadly, the finding that training dynamics encode a model's prior exposure opens new directions for auditing beyond contamination detection, including undisclosed fine-tuning or data poisoning, wherever the core question is what a model has seen rather than what it can do.

2 Test Set Contamination Detection Framework

Threat Model. We consider a third-party auditor who has white-box access to model weights and can perform fine-tuning on the suspect model. The auditor needs access to the benchmark of interest but does *not* need to know the suspect model's original training recipe, as we show that classifiers transfer across optimizers, learning rates, and model sizes (§4.3).

2.1 Definitions

The core idea is to compare two types of models: those that have been exposed to a target benchmark $\mathcal{B}$ during mid-training (*poisoned*) and those that have not (*clean*). We probe both by fine-tuning on $\mathcal{B}$ via SFT and ask whether the resulting loss trajectories are distinguishable.

To construct controlled comparisons, we start from a base model $M^{(0)}$ and a general mid-training dataset $\mathcal{D}$ with $\mathcal{B} \cap \mathcal{D} = \emptyset$. The *poisoning ratio* $p \in (0, 1]$ is the fraction of mid-training data consisting of benchmark samples. We create matched pairs:

- *Poisoned*: $M^{(1)}_{\text{poison}}$ trained on $\mathcal{B} \cup \mathcal{D}'$ where $|\mathcal{D}'| = |\mathcal{B}|/p - |\mathcal{B}|$, so benchmark data constitutes fraction p.
- *Clean*: $M^{(1)}_{\text{clean}}$ trained on $\mathcal{D}'' \subseteq \mathcal{D}$ with $|\mathcal{D}''| = |\mathcal{B}|/p$, matching the total sample count.

The *spurious performance augmentation* $\Delta := \text{Perf}(M^{(1)}_{\text{poison}}, \mathcal{B}) - \overline{\text{Perf}}(M^{(1)}_{\text{clean}}, \mathcal{B})$ measures how much contamination inflates benchmark scores, where the second term averages over the clean runs for that benchmark to reduce variance. We probe each $M^{(1)}$ by fine-tuning on $\mathcal{B}$ for one epoch via SFT, recording the per-step loss trajectory $\ell = (\ell_1, \ldots, \ell_T)$, which is the sole input to our classifier. Classification requires a *reference library* of such trajectories from known-clean and deliberately-poisoned models; our held-out-benchmark and cross-model evaluations show that this library can transfer to unseen benchmarks and suspect models of different architectures (§4.1).

Task Formulation. Given a collection of loss trajectories with known labels, we address two tasks: (1) *binary classification*, determining whether a model is clean or poisoned, and (2) *severity regression*, predicting the continuous value Δ. For classification, we retain only poisoned models with $\Delta > \delta$ for a threshold $\delta \geq 0$, ensuring that poisoned labels correspond to contamination that measurably affects benchmark performance.

2.2 Loss-Shape Based Contamination Classifiers

Given the naturally low-resource setting, we use two non-parametric approaches that require no distributional assumptions: (1) k-NN classification via majority vote, and (2) k-medoids clustering (Partitioning Around Medoids, PAM), which selects actual trajectories as cluster representatives, providing interpretable exemplars of clean versus contaminated learning dynamics. For regression, we adapt k-NN to predict Δ via distance-weighted averaging.

Both approaches rely on a pairwise distance metric between loss trajectories. Because loss trajectories are variable-length time series, Euclidean distance is inapplicable. We use Dynamic Time Warping (DTW) (Sakoe & Chiba, 1978), a widely used and empirically strong distance measure for time series classification (Ding et al., 2008) that elastically aligns two sequences to find the minimum-cost monotonic matching, naturally handling temporal shifts. See full formulation in Appendix A.4.

3 Experiments

3.1 Contamination Setups

After pre-training, modern LLMs undergo multi-stage training that typically includes mid-training and post-training (e.g., DPO). Contamination can enter at any stage, and an auditor must know whether the detection signal survives subsequent training and whether post-training contamination is detectable. We design three experiments of increasing pipeline complexity. All three evaluate SFT probing in Stage 2; Experiments 2 and 3 additionally evaluate DPO probing (Figure 2).

Experiment 1: Detecting Mid-Training Contamination without Post-Training. This is the base case. *Stage 1:* Create $M^{(1)}_{\text{clean}}$ and $M^{(1)}_{\text{poison}}$ via SFT on $\mathcal{D} \pm \mathcal{B}$, emulating mid-training. *Stage 2:* Fine-tune both on $\mathcal{B}$ via SFT and record loss trajectories. This experiment validates the core detection signal across all six models, 44 benchmarks, and the full hyperparameter grid.

Experiment 2: Detecting Mid-Training Contamination after Clean Post-Training. We investigate whether mid-training contamination remains detectable after subsequent clean post-training that could otherwise conceal the signal. *Stage 1a:* SFT on $\mathcal{D} \pm \mathcal{B}$, emulating mid-training (as in Experiment 1). *Stage 1b:* DPO on a clean preference dataset (Dolci (Team Olmo et al., 2025)) for *both* conditions—the benchmark never appears in post-training. *Stage 2:* Probe the resulting models by fine-tuning on $\mathcal{B}$ via SFT or DPO. This experiment tests whether contamination fingerprints introduced during mid-training persist through subsequent post-training.

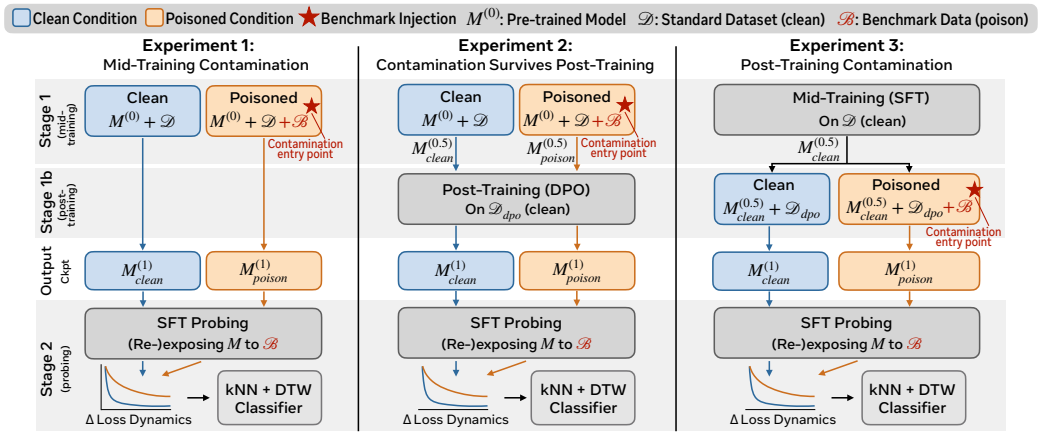


Figure 2: Contamination setups, differing in where benchmark $\mathcal{B}$ is injected. The diagram shows the shared SFT probing path in Stage 2; Experiments 2 and 3 additionally evaluate DPO probing. Classification uses only the resulting loss trajectory shape.

Experiment 3: Detecting Post-Training Contamination. This setting detects contamination that enters during post-training stage only. *Stage 1a:* Clean SFT on $\mathcal{D}$ for both conditions (identical checkpoints). *Stage 1b:* DPO on Dolci $\pm\,\mathcal{B}$—the poisoned model sees benchmark data as preference pairs during post-training. *Stage 2:* Probe using SFT or DPO on $\mathcal{B}$. This tests whether contamination introduced through preference optimization, which operates on relative rankings rather than direct next-token prediction, leaves a detectable trace in loss trajectories.

3.2 Experimental Setup

Benchmarks. We use 44 benchmarks from lm-eval-harness (Sutawika et al., 2024), each containing at least 1,000 samples. All benchmarks are either multiple-choice or short-form text generation where the lowest-perplexity option is taken as correct, and all are scored with accuracy $\in [0, 1]$. They cover a broad range of domains including natural language inference, commonsense reasoning, medical knowledge, social bias, and cybersecurity (full list in Appendix A).

Models. We use six models spanning three families and 1B–7B parameters: OLMo-2-0425-1B, OLMo-2-1124-7B (Walsh et al., 2025), Qwen2.5-1.5B, Qwen2.5-3B, Qwen2.5-7B (Qwen et al., 2024), and Llama-3.2-3B (Meta AI, 2024a). We prioritized developer diversity and similar release timing over release recency, since our experiments introduce the poisoning interventions ourselves; matching release times also lets us confirm some benchmarks postdate all models' training cutoffs. Experiment 1 uses all six models; Experiments 2 and 3 use OLMo-2-0425-1B with AdamW and Muon optimizers across 22 benchmarks. We separately evaluate cross-scale transfer to Llama-3.3-70B-Instruct (Meta AI, 2024b), collecting loss-only trajectories under ZeRO-3 on 32 H100 GPUs; this scaling study is not included in the six-model grid or its 43,000-run count.

Training Protocol. We emulate mid-training using SFT, computing loss from both prompts and completions, with Tulu-v2 (Ivison et al., 2023) as general dataset $\mathcal{D}$. Stage 1 trains $M^{(0)}$ for 1 epoch across learning rates $\in$ {1e-5, 5e-6, 1e-6, 5e-7}, two optimizers (AdamW, Muon), and two LR schedulers (constant, cosine). Stage 2 probing fine-tunes on the target benchmark at a fixed configuration (learning rate 1e-5, 1 epoch, batch size 8), subsampling each loss trajectory to 64 steps. Where applicable, Stage 1b post-training uses the Dolci preference dataset (Team Olmo et al., 2025).

Relevance Filtering. Not all poisoned models exhibit meaningful benchmark score inflation. When contamination has no measurable effect on performance ($\Delta \approx 0$), the poisoned label is misleading: the model behaves identically to a clean model despite the contaminated label. We therefore filter to runs where contamination produces a minimum performance gain of $\Delta \geq 0.05$, ensuring that poisoned labels correspond to contamination that practically affects benchmark scores.

Classifier Training and Evaluation. Unless otherwise stated, all experiments train and evaluate the kNN classifier on loss trajectories from a *single* model architecture, pooling all benchmarks and

Table 1: Within-model contamination detection using kNN+DTW ($\Delta \geq 0.05$). Each row pools all poisoning ratios, hyperparameters, and benchmarks for that model. 20-fold cross-validation split by benchmarks. Mean accuracy± std.

Model	Best k	Accuracy
OLMo2 1B	5	0.819 ± 0.148
Qwen2.5 1.5B	7	0.765 ± 0.166
Qwen2.5 3B	5	0.786 ± 0.148
Llama-3.2 3B	7	0.886 ± 0.126
Qwen2.5 7B	7	0.825 ± 0.162
OLMo2 7B	5	0.915 ± 0.122

Table 2: Cross-model contamination detection (Experiment 1, $\Delta \geq 0.05$). Each cell trains a kNN+DTW classifier on all trajectories from the row model and evaluates on the column model. $k = 8$. All poisoning ratios, hyperparameters, and benchmarks pooled. Mean accuracy: 87.0%.

	Evaluation Model					
Train Model	OLMo2 1B	Qwen 1.5B	Qwen 3B	Llama 3B	Qwen 7B	OLMo2 7B
OLMo2-1B	–	.833	.817	.847	.819	.900
Qwen-1.5B	.865	–	.921	.873	.895	.921
Qwen-3B	.865	.895	–	.887	.902	.912
Llama-3B	.875	.846	.855	–	.877	.913
Qwen-7B	.797	.826	.893	.843	–	.890
OLMo2-7B	.861	.841	.865	.891	.875	–

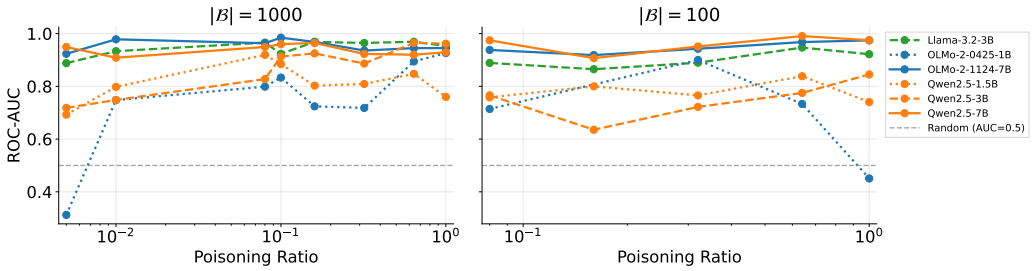


Figure 3: ROC-AUC scores for full benchmark contamination detection at varying poisoning ratios (Experiment 1, $\Delta \geq 0.05$). For each data point, a separate 8NN+DTW classifier is trained on a balanced dataset of clean and contaminated trajectories at that specific poisoning ratio for that model. 20-fold cross-validation split by benchmarks. See classification accuracy scores in Figure 7.

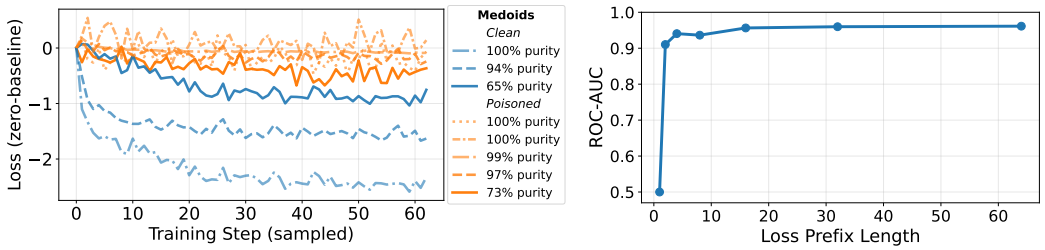
hyperparameter configurations. We use cross-validation with folds split by benchmark identity; the classifier never sees trajectories from evaluation benchmarks during training.

Baselines. Our task is *dataset-level* classification of models' learning behavior (i.e., does the contamination impact the model?), not per-sample membership inference. We adapt five MIA methods to our setting by computing per-sample scores across the benchmark, aggregating them into a single score per model, and classifying with logistic regression: (1) *Loss* (Yeom et al., 2018), (2) *Zlib* (Carlini et al., 2021), (3) *Min-K% (10%)* and (4) *Min-K% (20%)* (Shi et al., 2024), and (5) *Reference model ratio*, which computes the ratio of the suspect model's loss to a reference model's loss for each sample. Each baseline uses the same 20-fold cross-validation split by benchmark, and the same class balancing, as our classifier.

4 Results

4.1 Detecting Contamination During Mid-Training

Loss Trajectories Reliably Signal Contamination. Across six models and 44 benchmarks, FIRE achieves 77–92% detection accuracy at contamination levels that meaningfully inflate scores ($\Delta \geq 0.05$), with larger models generally yielding stronger detection (Table 1; models are trained jointly across all poisoning ratios with 20-fold cross-validation split by benchmark, and k is selected per model). Figure 3 breaks down performance by poisoning ratio for $|\mathcal{B}| \in \{100, 1000\}$, showing that 8NN+DTW remains strong across all values of $p \in [0.005, 1.0]$ considered. We report ROC-AUC alongside accuracy as it captures the full trade-off between true and false positive rates across all thresholds. Metrics are stable across poisoning ratios, so the signal should extend below the smallest p we test. OLMo-2 1B is the exception, with higher variance across ratios. Interestingly, detection performance also improves with model size: for both families with multiple sizes (OLMo-2 and Qwen2.5), the 7B variants consistently outperform their smaller counterparts.



(a) 8-medoids clustering

(b) Prefix predictive power (8NN+DTW)

Figure 4: Insights into distinct contamination patterns in loss trajectories. (a) 8-medoids clustering by contamination status; class purity shows percentage of cluster elements matching the medoid's true label. (b) ROC-AUC as a function of loss trajectory prefix length ($|\mathcal{B}| = 1000$). Shown for a representative cross-model setting (trained on Llama-3.2-3B, evaluated on OLMo2-7B); the early-prefix sufficiency result is consistent across model pairs.

Detection Transfers Across Model Architectures. An auditor may not have access to a reference library built from the same model architecture. Table 2 tests this by training a kNN+DTW classifier on all trajectories from one model and evaluating on a different model. Cross-model accuracy ranges from 80% to 92%, with a mean of 87.0% across all 30 model pairs. Strong transfers include Qwen2.5-1.5B $\rightarrow$ Qwen2.5-7B (89.5%) and the cross-family Qwen2.5-1.5B $\rightarrow$ OLMo2-7B setting (92.1%). Smaller-to-larger model transfers perform comparably to the reverse, meaning cheaper models can serve as reference classifiers for auditing larger models.

Detection Transfers to Models an Order of Magnitude Larger. On loss-only trajectories from Llama-3.3-70B-Instruct, classifiers trained on individual 1B–7B models achieve 69.2–77.7% zero-shot accuracy. Pooling the smaller models yields 66.5% because of a scale-induced bias toward predicting contamination, but adding only 50 labeled 70B trajectories raises accuracy to 88.7%, and 200 yield 98.2%. Appendix A.3 reports the full results and the ZeRO-3 setup on 32 H100 GPUs.

Training Dynamics Outperform Membership Inference Baselines. Figure 5 compares FIRE against five MIA baselines, all evaluated under the same $\Delta \geq 0.05$ filtering. FIRE outperforms all MIA baselines on three of six models and matches the best on the remaining three. The strongest baseline, reference model ratio (82% aggregated), requires a clean reference model of the same architecture—an assumption rarely satisfied in realistic auditing scenarios. FIRE achieves 77–92% without an architecture-matched reference model, which no baseline does. Static loss-based methods (Simple Loss, Min-K%, Zlib) perform near chance (57–61%), confirming the contamination signal resides in temporal dynamics rather than static snapshots. The two tasks differ: MIA answers a per-sample question of whether a specific example was leaked, while contamination detection is a dataset-level judgment of whether a benchmark as a whole influenced training. A static loss snapshot carries little of that dataset-level signal, which is what the near-chance numbers show.

4.2 Characterizing the Detection Signal

Trajectory Shape, Not Loss Level, Drives Detection. A natural hypothesis is that contaminated models simply start with lower loss on the benchmark, making detection trivial. However, all trajectories are zero-normalized before classification: each curve is shifted so that the initial loss is zero, removing absolute loss-level information entirely. DTW therefore operates solely on the *shape* of how loss changes over training steps. That shape alone achieves 77–92% accuracy at relevant contamination levels (Table 1), meaning the temporal dynamics of re-learning encode contamination independently of where the model starts.

Contamination Leaves an Early and Distinct Imprint on Loss Trajectories. Figure 4a illustrates the interpretability of FIRE: 8-medoids clustering (acc=0.924, AUC=0.969) produces medoids —actual trajectories from the dataset that serve as cluster representatives— offering concrete exemplars of clean versus contaminated dynamics. Clean medoids (shown in blue) exhibit sharp initial loss drops characteristic of a model encountering unfamiliar data, while contaminated medoids (shown in orange) show more gradual decay. The high cluster purity confirms these are genuinely distinct trajectory types rather than heavily overlapping distributions. Crucially, Figure 4b shows

Table 3: Aggregate detection accuracy under partial contamination ($\Delta \geq 0.05$). AdamW optimizer only for fair comparison across settings. Best k per model, 20-fold CV by benchmark.

Model	$\lvert\mathcal{B}\rvert = 1000$	Known $\lvert\mathcal{B}'\rvert = 100$	Unknown $\lvert\mathcal{B}'\rvert = 100$
OLMo2 1B	.839	.680	.669
Qwen2.5 1.5B	.859	.794	.698
Qwen2.5 3B	.874	.715	.735
Llama 3.2 3B	.925	.846	.831
Qwen2.5 7B	.850	.857	.850
OLMo2 7B	.871	.870	.900

Table 4: Cross-domain generalization accuracy for Experiment 1. Best accuracy across k values for kNN+DTW classifiers trained on one domain and evaluated on another ($\Delta \geq 0.05$).

Cross-Domain Type	Accuracy
Model Size	.890
Poison Ratio	.858
Stage-1 LR	.837
Optimizer	.793

that the discriminative signal concentrates in the early phase of fine-tuning, when a clean model undergoes rapid adaptation to novel data while a contaminated model traverses a qualitatively different path through loss space: ROC-AUC exceeds 0.95 within just 16 loss points and plateaus thereafter, corresponding to about 25% of a 1000-sample benchmark. This has direct practical implications: an auditor can detect contamination using a small fraction of the benchmark, reducing both compute cost and the amount of benchmark data that must be disclosed (Appendix A.1).

4.3 Detection Signal Generalizes Across Hyperparameters and Optimizers

We test whether classifiers built under one set of training conditions transfer to others, across four dimensions: (1) *cross-poisoning-ratio*, (2) *cross-model-size*, (3) *cross-learning-rate*, and (4) *cross-optimizer*. Table 4 summarizes results for Exp. 1. Cross-model-size transfer is the most robust (89%), followed by cross-poisoning-ratio (86%) and cross-learning-rate (84%) . Cross-optimizer transfer is the most challenging (79%) , reflecting fundamental differences in how optimizers shape loss landscapes. Each value is the mean over the transfer directions of that type, reported individually in App. Table 10. Informative asymmetries emerge in the transfer directions. For learning rates, low→high (86%) and high→low (82%) transfer comparably, a gap of 4 points. For optimizers, Muon→AdamW (87%) outperforms AdamW→Muon (71%) by 16 points. Full directional results in App. A.8.

4.4 Contamination Detection Does Not Require Knowing Exact Leaked Subset

In practice, an auditor may not know which benchmark examples were leaked. We evaluate this threat model by sampling two independent size-100 subsets from a benchmark $\mathcal{B}$ with $|\mathcal{B}| = 1000$: $\mathcal{B}' \subset \mathcal{B}$ is used to poison the model during Stage 1, and $\mathcal{B}'' \subset \mathcal{B}$ is drawn with a different seed for Stage 2 probing. The two subsets overlap only partially by chance (≈10 items in expectation), so the auditor probes with data largely disjoint from what was actually leaked, though drawn from the same distribution. Table 3 also reports the known-subset setting where $\mathcal{B}' = \mathcal{B}''$ as an upper bound.

Table 3 shows that unknown-subset detection accuracy is remarkably close to known-subset accuracy for most models. For models ≥3B, detection remains strong: OLMo-2-7B achieves 90.0%, Qwen2.5-7B 85.0%, and Llama-3.2-3B 83.1%. Smaller models show more variance (Qwen2.5-3B: 73.5%, Qwen2.5-1.5B: 69.8%, OLMo-2-0425-1B: 66.9%). The minimal degradation from known to unknown subsets suggests that FIRE detects a global contamination signal in the loss dynamics, not just leakage of specific items, making it robust even when the auditor cannot identify the exact leaked examples. More broadly, this sensitivity to global training history suggests that loss trajectories under controlled re-exposure may serve as a general probe of a model's prior data exposure, with potential applications to out-of-domain detection and dataset auditing beyond contamination.

4.5 Detection Under Adversarial Concealment

Regularization that keeps a model from becoming overconfident also dampens the memorization signature our classifier reads, whether or not the developer intends it. We test this *passive adversary* by training using label smoothing ($\epsilon = 0.1$) during Stage 1 poisoning. Table 5 shows that FIRE remains the strongest detector under label smoothing despite reducing its accuracy by 2.9 points,

Table 5: Detection accuracy when applying label smoothing during Stage 1 poisoning. Poisoning settings comprise 44 benchmarks over poisoning ratios 0.32 and 1.0, two learning rates, $k = 7$, for a total of 176 OLMo-2-0425-1B training runs. 15-fold cross-validation split by benchmark.

Setting	kNN+DTW (ours)	Simple Loss	Min-K%	Zlib
Standard	**71.2**%	58.6%	57.6%	57.2%
Label smoothing ($\epsilon = 0.1$)	**68.3**%	60.0%	61.7%	58.6%

holding a $\sim$7 point lead over every MIA baseline, all of which stay near chance in both conditions. Notably, label smoothing retained a high accuracy inflation ($+14.9\%$).

An *active adversary* that aims to directly optimize against the detector faces a more difficult computational problem. The detection signal is a full Stage 2 loss trajectory controlled by the auditor, not a static model property, so the adversary faces a bi-level optimization in which each poisoning update in the outer loop requires simulating a complete Stage 2 fine-tuning run in the inner loop. For a 7B model this is an $\sim 850\times$ cost multiplier over the poisoning run itself, roughly 1,700 GPU-hours of evasion gradients against 2 GPU-hours of poisoning. The auditor also chooses the Stage 2 learning rate, seed and optimizer, so the target is not fixed in advance. Detection costs one fine-tuning run; evasion costs a fine-tuning run inside every training step.

4.6 On Trajectory Shape as a Predictor of Performance Inflation

Loss trajectories can also predict the *magnitude* of spurious performance gains Δ, enabling quantitative assessment beyond binary detection. We train 8NN+DTW regressors with Mean Absolute Error (MAE) as the evaluation metric. Table 9 shows MAE of $\approx$0.11–0.17 when predicting only on contaminated models, a more challenging setting than including clean models (where MAE drops to 0.05–0.09). Figure 8 shows the residual distribution for a cross-model setting (trained on OLMo2-1B, evaluated on Qwen2.5-3B): the mean residual is -0.022, indicating a slight tendency to underestimate contamination severity, with most errors concentrated near zero. Appendix A.7 provides extensive cross-model evaluation with MAE 0.02–0.07 for mixed settings.

4.7 Contamination Persistence Through Post-Training

Using the pipelines described in §3.1, we investigate (1) whether mid-training contamination persists through subsequent post-training, and (2) whether post-training contamination is itself detectable.

Mid-Training Contamination Remains Detectable after Clean Post-Training (Experiment 2). We poison models during mid-training (Stage 1a), then apply clean post-training via DPO (Stage 1b), and probe with Stage 2 fine-tuning on $\mathcal{B}$. Table 11 shows that mid-training contamination remains detectable after post-training at 84.4% accuracy when probing with SFT (AdamW), comparable to Experiment 1's (mid-training only experiment) best results. DPO-based probing is less effective (57.4%), suggesting that the probing method matters: SFT fine-tuning re-activates the same learning signal that encoded the contamination, while DPO probing accesses it more indirectly through preference comparisons.

Mid-Training and Post-Training Encode Contamination Differently (Experiment 3). With mid-training held clean for both conditions, we poison during post-training via DPO (Stage 1b). An informative pattern emerges: DPO-stage contamination is detectable when probed with the matching training objective (DPO probing: 62.0% AdamW, 61.1% Muon), but SFT probing fails to recover the signal (46.5% AdamW, 50.0% Muon). The contrast with Experiment 2 tracks the training objective: each stage's contamination is recoverable by the objective that introduced it, and not by the other. SFT directly optimizes cross-entropy loss on training tokens, creating strong parameter-level imprints that persist even after post-training alters the model's behavior—hence the robust 84.4% detection in Experiment 2. DPO optimizes relative preferences between response pairs, leaving traces that are shallower but still accessible when the probing method matches the contamination pathway. This finding has practical implications: for multi-stage pipelines, mid-training contamination is the primary detection target, as it produces the strongest and most persistent signal. It also suggests that probing with the same training objective used for contamination can improve detection sensitivity.

5 Related Work

Test Set Contamination Detection. Early controlled studies distinguish mere memorization from *exploitation*, where contamination measurably improves task performance (Magar & Schwartz, 2022); this motivates our focus on contamination that produces positive spurious performance augmentation. Black-box detectors test exchangeability through canonical versus shuffled example orderings (Oren et al., 2024), compare confidence on benchmark items and distribution-matched counterparts (Zhang et al., 2024), or use behavioral probes such as the Data Contamination Quiz (Golchin & Surdeanu, 2025) and "Time Travel" prompts (Golchin & Surdeanu, 2024). Other methods exploit output-distribution statistics (Yang et al., 2023; Dong et al., 2024) or white-box internal activations (Liu et al., 2024). Surveys organize this literature across granularities (Xu et al., 2024) and show that common detection assumptions can fail under distribution shift (Fu et al., 2025). Most such methods extract a static score from a trained model; recent optimization-aware work instead detects individual pretraining examples from gradient magnitude, location, and concentration (Zhang et al., 2026). We likewise intervene on the model, but use the *entire dataset-level loss trajectory* induced by controlled re-exposure, rather than per-example activations, gradients, or forward-pass scores.

Membership Inference Attacks. Most MIAs infer per-example membership from static model behavior, via confidence/loss thresholding (Shokri et al., 2017; Yeom et al., 2018), likelihood-ratio tests against shadow models (LiRA (Carlini et al., 2022), RMIA (Zarifzadeh et al., 2024)), or token-probability statistics (Min-k% (Shi et al., 2024), Min-k%++ (Zhang et al., 2025)). Neighborhood comparison (Mattern et al., 2023) and semantic perturbations (Mozaffari & Marathe, 2024) improve robustness, though realistic LLM regimes drive many attacks toward random guessing (Duan et al., 2024). Dataset-level self-comparison is an important exception, measuring likelihood changes after paraphrasing part of a candidate sequence without requiring matched non-members (Ren et al., 2025). Closest in target, LLM dataset inference aggregates per-example MIA scores into a dataset-level statistical test (Maini et al., 2024); it reads a trained model's static outputs, whereas we read how the model responds to controlled re-exposure. Trajectory-based attacks use intermediate vision checkpoints (Liu et al., 2022); related work approximates training data from model weights (Morris et al., 2025) or actively fine-tunes a model for per-sample inference (Yin et al., 2026). Our target is also dataset-level, but our signal is the full loss-time curve's shape during deliberate fine-tuning on a suspect benchmark, designed to detect whether exposure materially affected the model even when individual membership is weak.

RL, SFT and Memorization. Post-training choices shape memorization in complex ways. RLHF tends to reduce memorization relative to SFT, though SFT-memorized samples often persist through RL (Pappu et al., 2025). Reward models can also exploit response length rather than response quality (Singhal et al., 2024), motivating adaptive length-bias mitigation (Bu et al., 2025). Training-dynamics studies reveal stable token-level learning signatures across runs (Chang et al., 2024), and larger models memorize faster and forget less during pretraining (Tirumala et al., 2022). Recent audits show that RL gains on math benchmarks can be confounded by contamination (Wu et al., 2026), while a dedicated RL-stage detector uses self-critique to expose policy-collapse signatures (Tao et al., 2026). Conversely, brief GRPO training can conceal contamination signals and chain-of-thought contamination can defeat memorization-based detectors (Wang et al., 2026). Our controlled DPO experiments therefore establish persistence in one important post-training setting, complementing rather than implying robustness across all RL algorithms.

6 Conclusion

A model's training history leaves traces in its weights that static outputs can miss. By treating auditing as active intervention, we recover this signal from loss trajectories produced while fine-tuning a suspect model on a target benchmark. FIRE, a simple non-parametric classifier, achieves 77–92% detection accuracy across six models and 44 benchmarks with as few as 256 samples; the signal survives post-training and transfers across architectures. Transfer also extends to Llama-3.3-70B-Instruct, reaching 77.7% zero-shot accuracy and 88.7% with 50 labeled 70B trajectories. Because the signal depends on the response to gradient updates, optimizing it away costs far more than producing it: each poisoning update requires a full probing run to obtain the loss curve, while auditing costs a single run of the classifier. More broadly, controlled re-exposure offers a window into training history for auditing undisclosed fine-tuning and data poisoning. See full limitations statement in Appendix B.

AI USE STATEMENT

Generative AI tools were used to assist with literature discovery, bibliographic verification, copyediting, and LaTeX organization. The authors reviewed the resulting citations and manuscript changes and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Sections 3.1 and 3.2 specify the experimental designs, models, data, hyperparameters, and evaluation protocol. Appendix A provides the benchmark list, compute estimates, robustness analyses, and detailed results needed to reproduce the reported comparisons.

REFERENCES

Yuyan Bu, Liangyu Huo, Yi Jing, and Qing Yang. Beyond excess and deficiency: Adaptive length bias mitigation in reward models for RLHF. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3091–3098. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.169. URL https://aclanthology.org/2025.findings-naacl.169/.

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)*, pp. 2633–2650. USENIX Association, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022. doi: 10.1109/SP46214.2022.9833649. URL https://doi.org/10.1109/SP46214.2022.9833649.

Tyler A. Chang, Zhuowen Tu, and Benjamin K. Bergen. Characterizing learning curves during language model pre-training: Learning, forgetting, and stability. *Transactions of the Association for Computational Linguistics*, 12:1346–1362, 2024. doi: 10.1162/tacl_a_00708. URL https://aclanthology.org/2024.tacl-1.74/.

Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008. doi: 10.14778/1454159.1454226. URL https://doi.org/10.14778/1454159.1454226.

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 *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12039–12050. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.716. URL https://aclanthology.org/2024.findings-acl.716/.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? In *Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=av0D19pSkU.

Yujuan Fu, Ozlem Uzuner, Meliha Yetisgen, and Fei Xia. Does data contamination detection work (well) for LLMs? a survey and evaluation on detection assumptions. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5250–5271. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.291. URL https://aclanthology.org/2025.findings-naacl.291/.

Shahriar Golchin and Mihai Surdeanu. Time travel in LLMs: Tracing data contamination in large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2Rwq6c3tvr.

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. URL https://aclanthology.org/2025.tacl-1.37/.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a changing climate: Enhancing LM adaptation with Tulu 2. arXiv preprint arXiv:2311.10702, 2023. URL https://arxiv.org/abs/2311.10702.

Yiyong Liu, Zhengyu Zhao, Michael Backes, and Yang Zhang. Membership inference attacks by exploiting loss trajectory. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2085–2098. ACM, 2022. doi: 10.1145/3548606.3560684. URL https://doi.org/10.1145/3548606.3560684.

Zhenhua Liu, Tong Zhu, Chuanyuan Tan, Bing Liu, Haonan Lu, and Wenliang Chen. Probing language models for pre-training data detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1576–1587. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.86. URL https://aclanthology.org/2024.acl-long.86/.

Inbal Magar and Roy Schwartz. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 157–165. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-short.18. URL https://aclanthology.org/2022.acl-short.18/.

Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. LLM dataset inference: Did you train on my dataset? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.06443.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 11330–11343. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.719. URL https://aclanthology.org/2023.findings-acl.719/.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta AI Blog, 2024a. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Meta AI. Llama 3.3 70B Instruct. Model card, 2024b. URL https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct.

John X. Morris, Junjie Oscar Yin, Woojeong Kim, Vitaly Shmatikov, and Alexander M. Rush. Approximating language model training data from weights. arXiv preprint arXiv:2506.15553, 2025. URL https://arxiv.org/abs/2506.15553.

Hamid Mozaffari and Virendra J. Marathe. Semantic membership inference attack against large language models. arXiv preprint arXiv:2406.10218, 2024. URL https://arxiv.org/abs/2406.10218.

Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B. Hashimoto. Proving test set contamination in black-box language models. In *International Conference on Learning Representations*, 2024. URL https://iclr.cc/virtual/2024/poster/18904.

Aneesh Pappu, Billy Porter, Ilia Shumailov, and Jamie Hayes. Measuring memorization in RLHF for code completion. In *International Conference on Learning Representations*, 2025. URL https://iclr.cc/virtual/2025/poster/29527.

Qwen, 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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024. URL https://arxiv.org/abs/2412.15115.

Jie Ren, Kangrui Chen, Chen Chen, Vikash Sehwag, Yue Xing, Jiliang Tang, and Lingjuan Lyu. Self-comparison for dataset-level membership inference in large (vision-)language model. In *Proceedings of the ACM on Web Conference 2025 (WWW '25)*, pp. 910–920, 2025. doi: 10.1145/3696410.3714703. URL https://doi.org/10.1145/3696410.3714703.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978. doi: 10.1109/TASSP.1978.1163055. URL https://doi.org/10.1109/TASSP.1978.1163055.

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 *International Conference on Learning Representations*, 2024. URL https://iclr.cc/virtual/2024/poster/17381.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017. doi: 10.1109/SP.2017.41. URL https://doi.org/10.1109/SP.2017.41.

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. A long way to go: Investigating length correlations in RLHF. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2310.03716.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas Muennighoff, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Julen Etxaniz, Chris, Hanwool Albert Lee, Zdeněk Kasner, Khalid, LSinev, Jeffrey Hsu, Anjor Kanekar, KonradSzafer, and AndyZwei. EleutherAI/lm-evaluation-harness: v0.4.3. Zenodo software release, 2024. URL https://zenodo.org/records/12608602.

Yongding Tao, Tian Wang, Yihong Dong, Huanyu Liu, Kechi Zhang, Xiaolong Hu, and Ge Li. Detecting data contamination from reinforcement learning post-training for large language models. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10010649.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3. arXiv preprint arXiv:2512.13961, 2025. URL https://arxiv.org/abs/2512.13961.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems 35*, pp. 38274–38290, 2022. doi: 10.52202/068431-2773.

URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/fa0509f4dab6807e2cb465715bf2d249-Abstract-Conference.html.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

Han Wang, Haoyu Li, Brian Ko, and Huan Zhang. On the fragility of benchmark contamination detection in reasoning models. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10008591.

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. In *AAAI Conference on Artificial Intelligence*, 2026. URL https://arxiv.org/abs/2507.10532.

Cheng Xu, Shuhao Guan, Derek Greene, and M-Tahar Kechadi. Benchmark data contamination of large language models: A survey, 2024. URL https://arxiv.org/abs/2406.04244.

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

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282. IEEE, 2018. doi: 10.1109/CSF.2018.00027. URL https://doi.org/10.1109/CSF.2018.00027.

Junjie Oscar Yin, John X. Morris, Vitaly Shmatikov, Sewon Min, and Hannaneh Hajishirzi. Learning to detect language model training data via active reconstruction. arXiv preprint arXiv:2602.19020, 2026. URL https://arxiv.org/abs/2602.19020.

Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58244–58282. PMLR, 2024. URL https://proceedings.mlr.press/v235/zarifzadeh24a.html.

Huixuan Zhang, Yun Lin, and Xiaojun Wan. PaCoST: Paired confidence significance testing for benchmark contamination detection in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1794–1809. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.97. URL https://aclanthology.org/2024.findings-emnlp.97/.

Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. Min-K%++: Improved baseline for pre-training data detection from large language models. In *International Conference on Learning Representations*, 2025. URL https://iclr.cc/virtual/2025/poster/29228.

Ruiqi Zhang, Lingxiang Wang, Hainan Zhang, Zhiming Zheng, and Yanyan Lan. From unfamiliar to familiar: Detecting pre-training data via gradient deviations in large language models. arXiv preprint arXiv:2603.04828, 2026. URL https://arxiv.org/abs/2603.04828.

A Additional Experimental Details

A.1 Computational Cost

The auditing cost consists of a single fine-tuning run on the target benchmark to collect the loss trajectory. Table 6 reports wall-clock times on a single NVIDIA A100 80GB GPU for the Stage-2 probing run (1 epoch, batch size 8, no gradient accumulation).

Table 6: Wall-clock time for a single Stage-2 probing run (1 epoch on $|\mathcal{B}|$ samples, batch size 8, single A100 80GB). Classification via DTW+kNN adds $<$1 second.

Model	$\lvert\mathcal{B}\rvert = 1000$	$\lvert\mathcal{B}\rvert = 256$
OLMo-2-0425-1B	$\sim$2 min	$\sim$30 sec
Qwen2.5-1.5B	$\sim$3 min	$\sim$45 sec
Qwen2.5-3B / Llama-3.2-3B	$\sim$5 min	$\sim$1.5 min
Qwen2.5-7B / OLMo-2-1124-7B	$\sim$10 min	$\sim$3 min

Building the reference library of clean and contaminated trajectories is a one-time cost. For a single model and benchmark, this requires fine-tuning the model at several poisoning ratios (5 contaminated + 1 clean configuration), totaling approximately 6$\times$ the single-run cost. Once the reference library is built, auditing a new suspect model requires only one fine-tuning run plus a kNN lookup (under 1 second). The total compute for all 43,000+ runs in the core six-model study was approximately 2,000 A100 GPU-hours. A single 70B audit takes approximately 1 GPU-hour on an H100, negligible relative to training or even evaluating a model of that size. The separate 70B scaling study used ZeRO-3 across 32 H100 GPUs and is excluded from both that run count and the A100 timing estimates in Table 6.

A.2 Membership Inference Baselines

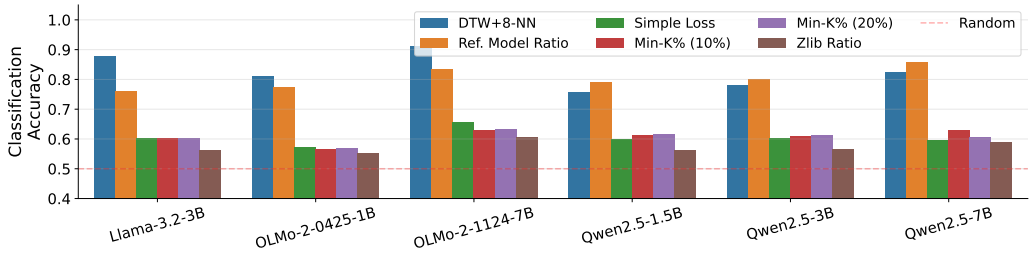


Figure 5: Per-model comparison of 8NN+DTW (ours) against five MIA baselines (Experiment 1, $\Delta \geq 0.05$). The dashed line marks chance.

A.3 Full 70B Scaling Results

Table 7 reports zero-shot transfer from each smaller model. Table 8 reports the effect of adding labeled in-scale trajectories to the 15,796 pooled 1B–7B training trajectories. The 3NN+DTW classifier is evaluated on the remaining 70B data, with mean and standard deviation calculated over 10 random splits.

A.4 DTW Formulation

For trajectories $\ell^{(i)} = (\ell_1^{(i)}, \ldots, \ell_{T_i}^{(i)})$ and $\ell^{(j)} = (\ell_1^{(j)}, \ldots, \ell_{T_j}^{(j)})$, the DTW distance is defined as $\mathrm{DTW}(\ell^{(i)}, \ell^{(j)}) := \sqrt{\min_\pi \sum_{(s,t)\in\pi} (\ell_s^{(i)} - \ell_t^{(j)})^2}$, where π is a warping path that aligns elements from both trajectories while satisfying monotonicity and boundary constraints. DTW is computed via dynamic programming as $D(s,t) = (|\ell_s^{(i)} - \ell_t^{(j)}|)^2 + \min\{D(s-1,t), D(s,t-1), D(s-1,t-1)\}$ with boundary conditions $D(1,1) = (|\ell_1^{(i)} - \ell_1^{(j)}|)^2$ and $D(s,0) = D(0,t) = \infty$. The final DTW distance is $\mathrm{DTW}(\ell^{(i)}, \ell^{(j)}) = \sqrt{D(T_i, T_j)}$.

Table 7: Zero-shot transfer to Llama-3.3-70B-Instruct.

Training source	Accuracy
OLMo2 1B	.700
Qwen2.5 1.5B	.754
Qwen2.5 3B	.777
Llama-3.2 3B	.752
Qwen2.5 7B	.692
OLMo2 7B	.735
All 1B–7B	.665

Table 8: Transfer after adding n_{70B} labeled 70B trajectories. Mean $\pm$ std over 10 random splits.

n_{70B}	Accuracy	Clean recall	Poisoned recall
0	.665 $\pm$ .000	.331	1.000
10	.710 $\pm$ .020	.421	.999
20	.771 $\pm$ .023	.557	.986
50	.887 $\pm$ .030	.790	.984
100	.944 $\pm$ .024	.911	.977
200	.982 $\pm$ .007	.989	.975
260	.988 $\pm$ .006	.992	.984

Benchmarks Throughout this paper we use 44 benchmarks, all hosted in lm-eval. They cover multiple-choice and short-form generation tasks. The full list is: `anli_r1`, `anli_r2`, `anli_r3`, `arc_easy`, `arc_eu_easy`, `bigbench_bbq_lite_json_multiple_choice`, `bigbench_color_multiple_choice`, `bigbench_cs_algorithms_multiple_choice`, `bigbench_fact_checker_multiple_choice`, `bigbench_hyperbaton_multiple_choice`, `bigbench_navigate_multiple_choice`, `bigbench_social_iqa_multiple_choice`, `bigbench_temporal_sequences_multiple_choice`, `bigbench_tracking_shuffled_objects_multiple_choice`, `bigbench_winowhy_multiple_choice`, `boolq`, `eus_exams_es_osakidetza6c`, `eus_trivia`, `histoires_morales`, `inverse_scaling_pattern_matching_suppression`, `mastermind_24_easy`, `mastermind_35_hard`, `mastermind_46_easy`, `mastermind_46_hard`, `medmcqa`, `medqa_4options`, `mnli`, `mnli_mismatch`, `moral_stories`, `multirc`, `persona_optionality-increasing`, `persona_optionality-preservation`, `persona_politically-conservative`, `persona_politically-liberal`, `persona_resource-acquisition`, `piqa`, `prost`, `qnli`, `qqp`, `race`, `swag`, `sycophancy_on_philpapers2020`, `wmdp_bio`, and `wmdp_cyber`.

A.5 Distributional Details on Clean and Poisoned Models

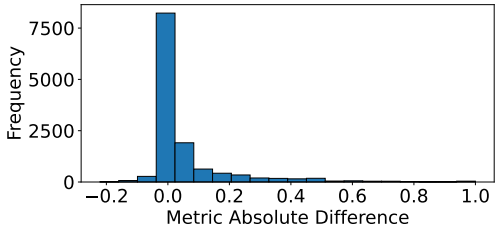


Figure 6: Histogram of spurious performance augmentation Δ values across all (model, benchmark, hyperparameter) configurations with $|\mathcal{B}| = 1000$. Many configurations produce $\Delta \approx 0$, motivating our relevance filtering ($\Delta \geq 0.05$). Higher poisoning ratios tend to produce larger Δ.

A.6 Extra Analyses for Benchmark Contamination

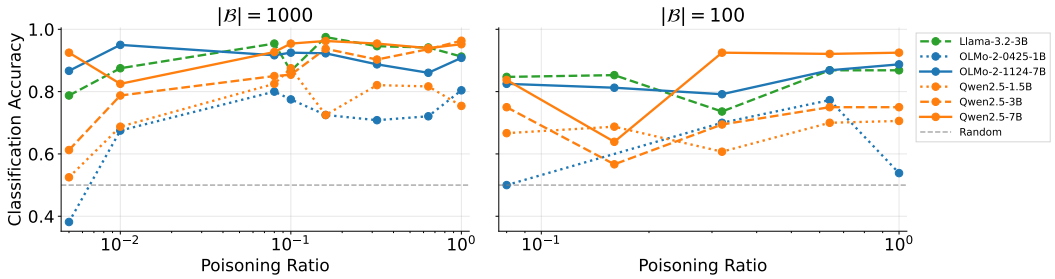


Figure 7: Classification accuracy for the setting of Figure 3, which reports ROC-AUC for the same classifiers.

A.7 Beyond Binary Classification

Table 9: Regression performance for predicting spurious performance gains Δ using 8NN+DTW for $|\mathcal{B}| = 1000$. MAE (lower is better), mean $\pm$ std across folds. "Poisoned" includes only contaminated models; "All" includes both.

Model	MAE (poisoned)	MAE (all)
OLMo-2-0425-1B	0.111 ± 0.068	0.047 ± 0.028
Qwen2.5-1.5B	0.165 ± 0.109	0.064 ± 0.039
Qwen2.5-3B	0.173 ± 0.098	0.074 ± 0.036
Llama-3.2-3B	0.143 ± 0.137	0.048 ± 0.038
Qwen2.5-7B	0.115 ± 0.100	0.055 ± 0.039
OLMo-2-1124-7B	0.170 ± 0.117	0.089 ± 0.059

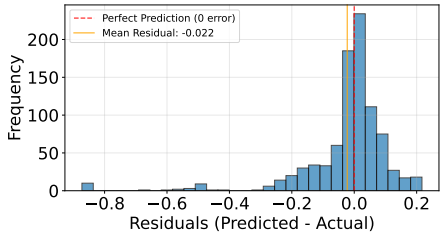


Figure 8: Residuals (predicted Δ minus actual Δ) for contaminated models. The 8NN+DTW regressor is trained on OLMo2-1B and evaluated on Qwen2.5-3B.

A.8 Detailed Cross-Domain Generalization Results

This section reports every directional transfer behind Table 4, whose entries are the mean over the directions listed here for each split type. Table 10 gives the full breakdown for Experiment 1.

Cross-Model-Size Details Every size transfer reaches at least 86.5%. Among single-source directions, medium→large (90.8%) is the strongest and small→medium (86.5%) the weakest, and pooling two size groups helps slightly (small+medium→large, 91.4%).

Cross-Poisoning-Ratio Details All directions fall between 82.1% and 89.1%. The best is medium→low (89.1%). Pooling source ratios (high+medium→low, 86.4%) improves on high→low (84.5%) but does not reach the best single source, so training on a single well-chosen ratio suffices.

Cross-Stage-1-LR Details Transfer from low to high learning rates (85.7%) exceeds the reverse (81.6%).

Cross-Optimizer Details Muon→AdamW transfer (87.2%) substantially outperforms AdamW→Muon (71.4%), making this the weakest transfer direction we observe.

A.9 Detailed Post-Training Contamination Results

Cross-Domain Results for Post-Training Experiments Cross-poisoning-ratio transfer remains robust in the post-training experiments (Table 12). Exp2 reaches 91.7% when transferring from

Table 10: Directional cross-domain transfer accuracy for Experiment 1 ($\Delta \geq 0.05$). Best k shown in parentheses. Table 4 reports the mean over the directions within each split type.

Transfer Direction	Accuracy
Cross-Model-Size	
small+medium → large	.914 (k=2)
medium → large	.908 (k=3)
large+medium → small	.900 (k=2)
small → large	.886 (k=3)
large → small	.867 (k=2)
small → medium	.865 (k=3)
Mean	*.890*
Cross-Poisoning-Ratio	
medium → low	.891 (k=2)
low → high	.879 (k=3)
high+medium → low	.864 (k=2)
high → medium+low	.853 (k=3)
high → medium	.850 (k=4)
high → low	.845 (k=4)
high($\geq$0.32)+clean → low($\leq$0.08)+clean	.821 (k=4)
Mean	*.858*
Cross-Stage-1-LR	
low(1e-6, 5e-7, 1e-7) → high(1e-5, 5e-6)	.857 (k=3)
high(1e-5, 5e-6) → low(1e-6, 5e-7, 1e-7)	.816 (k=2)
Mean	*.837*
Cross-Optimizer	
Muon → AdamW	.872 (k=3)
AdamW → Muon	.714 (k=8)
Mean	*.793*

Table 11: Post-training contamination detection accuracy (Experiments 2 and 3). Within-domain kNN+DTW classification with cross-validation by benchmarks. Best k shown. "Stage-2 Method" indicates whether probing uses SFT or DPO fine-tuning on $\mathcal{B}$.

Experiment	Optimizer	Stage-2 Method	Best k	Accuracy	n curves
Exp2: mid-training→post-training	AdamW	SFT	8	.844	161
	AdamW	DPO	3	.574	124
	Muon	SFT	5	.675	110
	Muon	DPO	2	.611	108
Exp3: Post-training poisoning	AdamW	SFT	8	.465	172
	AdamW	DPO	7	.620	154
	Muon	SFT	3	.500	68
	Muon	DPO	5	.611	58

high ratios and clean runs to low ratios, and 85.7% from high to medium. Exp3 reaches 80.8% from high to medium and low. Cross-optimizer transfer is weaker in both, at 64.5–71.2% for Exp2 and 55.8–60.0% for Exp3, consistent with cross-optimizer being the hardest transfer in Experiment 1.

Table 12: Cross-domain results for the post-training experiments ($\Delta \geq 0.05$). Best accuracy across k values.

	Exp2		Exp3	
Transfer Direction	Accuracy	Best k	Accuracy	Best k
Cross-Optimizer				
AdamW → Muon	.645	2	.600	3
Muon → AdamW	.712	7	.558	3
Cross-Poisoning-Ratio				
high → medium	.857	2	.800	5
high → medium+low	.857	2	.808	3
high($\geq$0.32)+clean → low($\leq$0.08)+clean	.917	3	.750	3

B LIMITATIONS

FIRE requires white-box access to model weights and the ability to fine-tune, which excludes closed API-only models. Logprob-based trajectory proxies via API access and extensions to multimodal settings are open directions. Our evaluation focuses on multiple-choice and short-form generation benchmarks; extending to free-form generation tasks where loss signals are noisier is an open question. Cross-optimizer transfer is the weakest form of generalization we observe, and our post-training experiments use a single model family, leaving broader scale validation and extension to RLHF, GRPO, and other post-training algorithms to future work. We also do not evaluate an adaptive adversary explicitly optimizing the training-dynamics signal, so resistance to such evasion remains an open empirical question. Probing model dynamics under controlled re-exposure may also apply to other auditing tasks, including detection of undisclosed fine-tuning and data poisoning.