LUCID: UNIVERSAL AUDITING OF DISTILLED LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Data distillation lets student models cheaply inherit capabilities from a stronger teacher, but it also creates a new form of misappropriation risk: a suspect model may be trained on a protected model's outputs, quietly transferring the teacher's proprietary capabilities while remaining architecturally unrelated. Existing fingerprinting techniques mostly target model-family attribution or rely on white-box access, offering little help when the suspect model is closed-source and only its responses can be observed. We introduce **LUCID** for auditing distilled-data use with only query access to the suspect model. LUCID inserts each suspect response into a self-attribution prompt, extracts its judge-token representation from the protected model, and trains a lightweight classifier on representations collected across multiple distillation settings. Our theoretical analysis characterizes risk variation and averaging across distillation environments. Extensive experiments varying suspect-model family and scale, protected-model scale, distillation prompt style, and audit question set show that LUCID consistently achieves high detection performance and generalizes robustly across diverse distillation and auditing settings.

1 INTRODUCTION

High-capacity large language models (LLMs) are increasingly used as teachers to synthesize chain-of-thought (CoT) data for fine-tuning compact student models, transferring reasoning skills through data rather than direct weight sharing. For example, Magister et al. (2023) show that students fine-tuned on teacher-generated CoT outputs improve on arithmetic, commonsense, and symbolic reasoning benchmarks. Likewise, Zhu et al. (2024) introduce Program-aided Distillation (PaD), which distills programmer-verified synthetic reasoning data into smaller models and outperforms conventional CoT fine-tuning.

These reasoning capabilities constitute a core part of a model's intellectual property (Figure 1, top-left). Safeguarding them requires protecting the distilled datasets generated by high-capacity models, particularly those tailored to specific domains. These datasets can be misappropriated for supervised fine-tuning (SFT) to transfer reasoning capabilities to other models (Figure 1, bottom-left). Unauthorized use of these data constitutes infringement of the protected model's intellectual property.

Unauthorized reuse of distilled data also raises privacy and safety concerns. Knowledge distillation can propagate or amplify privacy leakage: distilled students can be more vulnerable to membership inference attacks (Cui et al., 2025), and black-box queries can expose sensitive client information in federated distillation (Shi et al., 2025a). These concerns are particularly acute as LLMs acquire specialized capabilities in healthcare (Xie et al., 2024; Lin et al., 2025; Zhao et al., 2024), law (Shu et al., 2024; Colombo et al., 2024), and finance (Ke et al., 2025), where uncontrolled capability transfer increases the risk of misuse and unsupervised dissemination.

Detecting this transfer is challenging. Signature-based protections such as EmMark (Zhang & Koushanfar, 2024) require modifying the model in advance and therefore do not apply to already-released public models. Natural fingerprinting instead exploits patterns already present in model outputs (McGovern et al., 2024; Tang et al., 2023; Suzuki et al., 2025), but attributing a response to a particular model does not, on its own, establish whether another model was trained on that model's outputs. We discuss this distinction in Section 2.2 and probe the reliability of statistical similarity across distillation settings in Section 2.3.

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.

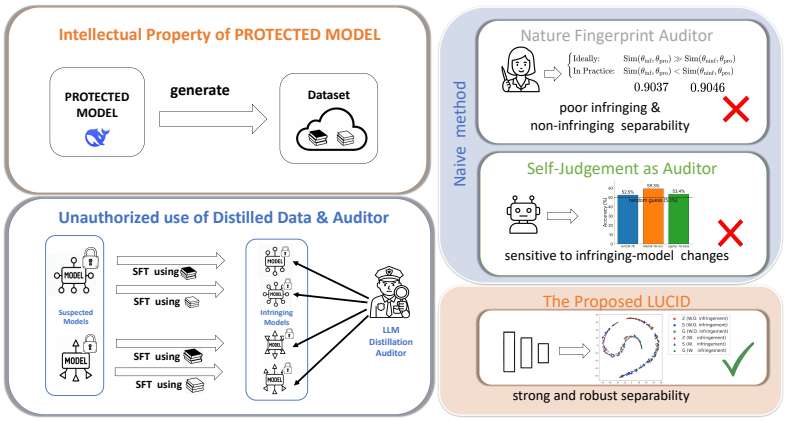


Figure 1: Overview of distilled-data infringement and the auditing pipeline. The auditor determines whether a suspect model was fine-tuned on data generated by the protected model. A suspect model is infringing if such data were used without authorization.

To address this problem, we use the protected model to evaluate responses from infringing and non-infringing models. We extract the internal representation of its judge token and train a classifier across diverse infringement settings. We analyze prediction risk across environments and empirically evaluate the classifier's ability to trace the use of protected-model outputs in unseen settings.

In summary, our contributions are as follows:

- **Problem formulation.** We formulate black-box auditing of distilled data in supervised fine-tuning and explain why existing fingerprinting methods are inadequate for this setting.
- **Auditing method.** We introduce LUCID, which audits suspect models through query access alone, using protected-model judge-token representations and achieving over 90% accuracy in all evaluated settings.
- **Generalization evaluation.** We validate generalization to unseen suspect families and prompt styles, and robustness across model scales, data mixtures, and audit question sets.

2 Preliminaries

Section 2.1 defines the auditor's access to the protected and suspect models. Section 2.2 then reviews proactive and natural fingerprinting approaches, and Section 2.3 uses a controlled experiment to examine why statistical similarities are unreliable for distillation auditing.

2.1 Auditor Capabilities and Scope

The **protected model** generates the data that the auditor seeks to safeguard, and the auditor has full access to its architecture, parameters, and internal representations. The **suspect model** is accessible only through queries: the auditor can observe its text responses but not its parameters, logits, or hidden states. Methods requiring internal features from the suspect model, such as REEF (Zhang et al., 2025a), therefore fall outside our setting. Moreover, an infringer may vary the prompts, sampling or optimization settings, and student model families used for distillation. An effective auditor must therefore remain reliable across diverse distillation settings.

2.2 Existing Auditing Approaches

Proactive Fingerprints. Proactive methods embed identifying signatures through training. UTF uses supervised fine-tuning on undertrained tokens to implant specific input–output pairs (Cai et al., 2024). ADFP perturbs tokens during response generation to embed a fingerprint (Xu et al., 2026), but

must be applied broadly since the owner cannot know which responses will be collected, and any unfingerprinted output remains unprotected. More generally, proactive methods require intervention in either the protected model or its generation process before distillation and may be targeted by fingerprint-removal techniques (Zhang et al., 2025b). They therefore do not directly address our setting, in which the auditor seeks to trace the use of data generated by an existing model without prior fingerprint insertion.

Natural Fingerprints. Output-based attribution methods exploit statistical characteristics of generated text, such as lexical patterns, without inserting any signature into the model (McGovern et al., 2024; Tang et al., 2023; Wadhwa et al., 2025; Huang et al., 2026). Suzuki et al. (2025) show that differences in training dynamics can leave distinguishable patterns in model outputs even when the training data are identical. This attribution signal, however, targets *which* model produced a response rather than whether the response has been reused for training a second model, and Section 2.3 shows that the corresponding statistical similarity is unstable in that setting. Appendix A.1 provides further related work.

2.3 Limitations of Natural Fingerprints

To examine natural fingerprinting in distillation auditing, we use DeepSeek-R1-Distill-Qwen-14B as the protected model θ_{pro} and construct the protected corpus from its responses to the `s1k` questions (Muennighoff et al., 2025). We fine-tune Seed-Coder-8B-Reasoning on this corpus to obtain the infringing model θ_{inf}, while GritLM-7B serves as the non-infringing model θ_{ninf}. On the same questions, the average top-10 word-frequency cosine similarity between θ_{inf} and θ_{pro} is 0.9037, lower than the 0.9046 similarity between θ_{ninf} and θ_{pro}. This reversal shows that surface-level response statistics do not reliably preserve data-provenance signals after distillation. The evaluation across four suspect families and three prompt styles in Section 4.2 further supports this finding, motivating an auditor that goes beyond vocabulary and generation-style similarities.

3 LUCID for Distillation Auditing

3.1 Self-Judgment as a Motivating Perspective

We next examine whether the protected model's evaluative capabilities provide a stronger auditing signal than surface-level statistics. The self-attribution prompt in Figure 2 asks the protected model to assess a candidate response using its syntax, word choice, and reasoning patterns. In the direct *Self-Judge* baseline, the resulting Yes/No answer serves as the auditing verdict.

> Given the following response {response}, determine if it was generated by you, focusing on the syntactic structure and textual patterns. Provide your reasoning based on the analysis of the sentence construction, word choice, and overall phrasing. Please respond with either 'Yes' or 'No' first. Was this answer generated by you? Answer:

Figure 2: Self-attribution prompt used to elicit the judge token.

We use DeepSeek-R1-Distill-Qwen-14B as the protected model and construct infringing variants of GritLM-7B, Mistral-7B-v0.1, and Zephyr-7B-beta by fine-tuning on its distilled `s1k` data. Figure 3 shows that direct self-judgment performs only slightly better than random guessing and varies substantially across suspect families.

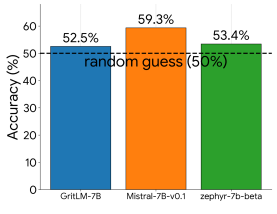
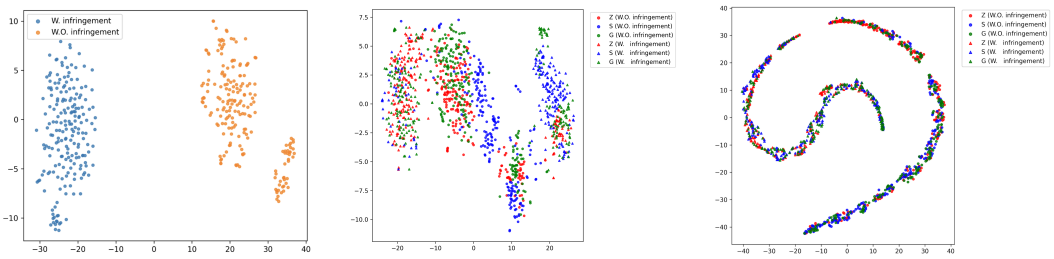


Figure 3: Self-Judge detection accuracy.

The failure case in Appendix A.2 helps explain this instability. Even when the protected model misclassifies a non-infringing response, **its analysis of word choice, linguistic structure, and content remains informative**, which motivates decoupling the model's representation of the response from its final verdict.



(a) Simple scenario. (b) Complex scenario w/o LUCID. (c) Complex scenario w/ LUCID.

Figure 4: t-SNE visualizations of judge-token representations in the simple setting (a) and the multi-family setting before classification (b), and of MLP penultimate-layer representations after training (c).

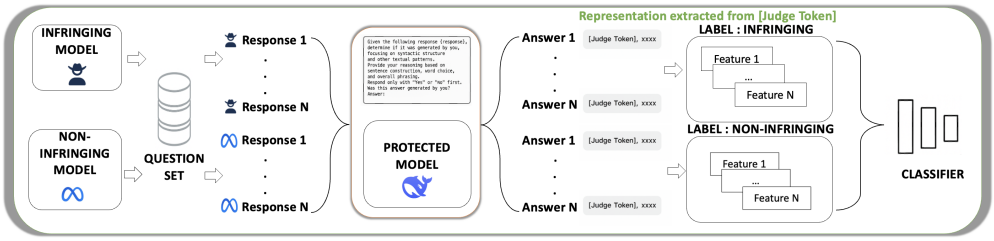


Figure 5: Overview of the LUCID training procedure. For responses generated by suspect models on a shared set of audit questions, we use the frozen protected model to obtain the hidden representation of the judge token. The resulting representations, together with their infringement labels, are used to train an MLP classifier.

Our investigation suggests that Self-Judgment fails because **LLMs lack a clear notion of authorship and therefore cannot reliably decide whether a given text was generated by themselves**, even though they can recognize patterns in their own behavior. Self-Judgment therefore cannot serve as a direct auditor, yet the strong understanding ability of LLMs still exposes useful intermediate features.

3.2 Judge-Token Representations for Distillation Auditing

Although direct self-judgment is unreliable, the protected model's internal representation of a suspect response remains informative. We insert the response into the self-attribution prompt and let the protected model generate a single *judge token*, and extract its hidden state from a chosen transformer layer. Rather than using the discrete verdict, LUCID feeds this representation to a trained classifier; token-level representations have been shown to preserve fine-grained cues useful for eliciting model beliefs (Vazhentsev et al., 2025).

We examine these representations by collecting responses from the non-infringing GritLM-7B model and an infringing model on the same GSM8K audit questions (Cobbe et al., 2021). We pass each response to the protected model using the prompt in Figure 2 and extract the judge-token hidden state. Figure 4a visualizes the resulting features: the two groups occupy distinct, linearly separable regions. This separation motivates training a classifier on judge-token representations to detect unauthorized distillation.

3.3 Multi-Factor Classification for Robust Auditing

The separation in Figure 4a is obtained in a simplified setting. In practice, suspect families, distillation prompts, and training choices all vary. To isolate the effect of model family, we hold the prompt format fixed and construct paired infringing and non-infringing variants of GritLM-7B, Seed-Coder-8B-Reasoning, and Zephyr-7B-beta. Figure 4b shows that this additional variation reduces separability: representations from an infringing model in one family can overlap with those from non-infringing models in other families.

To account for variation across distillation settings, we pool judge-token representations from multiple *environments*, each containing paired infringing and non-infringing variants under a given suspect family and training configuration. Motivated by invariant learning (Peters et al., 2016; Arjovsky et al., 2019), we analyze risk under an idealized latent model. Let $Z^e = g(Z_i, Z_s^e)$, with a shared representation map g, an invariant latent $Z_i \sim P_i$, and an environment-specific nuisance $Z_s^e \sim P_s^e(\cdot \mid Z_i)$. Assume a common label law $P(Y \mid Z_i)$ and $Y \perp Z_s^e \mid Z_i$ in every environment. For a fixed classifier f and loss $0 \le \ell \le 1$, define $\mathcal{L}^e(f) = \mathbb{E}_{(Z^e,Y)}\ell(Y, f(Z^e))$. These bounds concern bounded auditing losses, such as 0–1 error; the classifier is trained with cross-entropy.

Theorem 3.1. *Under the latent model above, let* $D_{e,t} = \mathbb{E}_{Z_i \sim P_i} D_{\mathrm{KL}}\big(P_s^e(\cdot \mid Z_i) \| P_s^t(\cdot \mid Z_i)\big)$. *If* $D_{e,t} < \infty$, *then*

$$\big|\mathcal{L}^e(f) - \mathcal{L}^t(f)\big| \le \sqrt{D_{e,t}/2}. \tag{1}$$

Theorem 3.1 bounds risk variation by conditional nuisance shift. If the nuisance is independent of Z_i in both environments, $D_{e,t}$ reduces to the KL divergence between their nuisance marginals. We next quantify concentration of the average risk across sampled environments.

Theorem 3.2. *Let* $e_1, \ldots, e_M \overset{\text{i.i.d.}}{\sim} \pi$, *and fix* f *independently of these draws. Define* $\overline{\mathcal{L}}_M(f) = M^{-1}\sum_{j=1}^M \mathcal{L}^{e_j}(f)$ *and* $\mathcal{L}_\pi(f) = \mathbb{E}_{e\sim\pi}\mathcal{L}^e(f)$. *For any* $\delta \in (0,1)$, *with probability at least* $1-\delta$,

$$\big|\overline{\mathcal{L}}_M(f) - \mathcal{L}_\pi(f)\big| \le \sqrt{\frac{\log(2/\delta)}{2M}}. \tag{2}$$

These results characterize population risks under the stated assumptions. Theorem 3.2 concerns a fixed classifier and an environment-averaged risk that retains nuisance variation; it does not establish nuisance removal by training. We evaluate the trained MLP's generalization to held-out settings empirically in Section 4.1. Proofs are provided in Appendix A.13.

3.4 The LUCID Pipeline

LUCID pools judge-token representations and infringement labels across suspect families and distillation settings to train an MLP, as illustrated in Figure 5.

LUCID Training. As illustrated in Figure 5, each infringing or non-infringing model first answers a designated audit question set. For each response, we prompt the protected model to generate a single "Yes" or "No" judge token and extract the hidden state of that token. This feature vector is fed into a multilayer perceptron (MLP) that outputs a two-dimensional infringement score. We minimize the binary cross-entropy loss

$$-\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log \sigma\big(f(\mathbf{z}_i)\big) + (1-y_i)\log\big(1-\sigma\big(f(\mathbf{z}_i)\big)\big)\Big],$$

where f is the MLP classifier, $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function, and $N = E \times m$ is the total number of training samples across E models each producing m responses. With feature dimension d, $\mathbf{Z} \in \mathbb{R}^{N\times d}$ stacks the per-response embeddings $\mathbf{z}_i$, and $\mathbf{Y} \in \{0,1\}^{N\times 1}$ stores the labels. The resulting classifier concentrates on the feature dimensions most relevant to detecting use of the distilled corpus. The protected model stays frozen throughout data generation and judge-token feature extraction; only the suspect models undergo SFT to construct the experimental checkpoints, and the MLP detector is trained separately on the extracted features. Complete settings are provided in Appendices A.3 and A.4.

LUCID Auditing. At test time, we collect the suspect model's responses and extract their judge-token representations from the protected model following steps 4 and 5 in Appendix A.3. The trained classifier is held fixed and applied to these features to determine infringement status. To visualize the learned decision boundary, Figure 4c plots the penultimate-layer embeddings of the MLP on a held-out test set of infringing and non-infringing checkpoints (built on GritLM-7B, Seed-Coder-8B, and Zephyr-7B) generated with sampling strategies disjoint from the training set. The features from infringing models form a single tight cluster, cleanly separated from the non-infringing cluster, showing class separation on these held-out checkpoints.

4 EXPERIMENTS

We evaluate robustness across prompt styles and suspect families in Section 4.1 and compare detection methods across protected-model sizes in Section 4.2. Section 4.3 extends the evaluation to larger suspect models. We then evaluate mixed-source distillation and publicly released models in Sections 4.4 and 4.5, respectively. Finally, Section 4.6 studies the effects of distilled-corpus size, extraction layer, and audit question set.

4.1 ROBUSTNESS ACROSS MODEL FAMILIES AND PROMPTS

To assess the generalization of LUCID, we designate DeepSeek-R1-Distill-Qwen-14B as the *protected model* and treat its responses to the `s1k` question set as the *protected corpus*. Following steps 1–3 in Appendix A.3, we build infringing and non-infringing suspects, query them with the same GSM8K problems, and insert each response into the self-attribution prompt of Figure 2 to extract the protected model's judge-token representation. We then vary two experimental factors.

Generalization across prompt styles. LUCID achieves an average accuracy of 99.5% across the three held-out prompt-style evaluations (Table 1). For each evaluation, we train the classifier on judge-token representations from two prompt styles and test it on suspect models distilled using the remaining style. We repeat this process for Original, Alpaca-style, and Reflection-style prompts. Prompt details are provided in Appendix A.8.

Generalization across model families. LUCID achieves an average accuracy of 99.6% when trained on three suspect families and evaluated on the held-out fourth (Table 2). We evaluate four families: GritLM-7B (G), Mistral-7B-v0.1 (M), Seed-Coder-8B-Reasoning (S), and Zephyr-7B-beta (Z). Together with Table 1, this result shows that LUCID generalizes to unseen prompt styles and suspect families.

Table 1: Detection accuracy (%) on a held-out distillation prompt style.

Prompt style	Orig.	Alpaca	Refl.
Accuracy	99.2	99.3	99.9

Table 2: Detection accuracy (%) on a held-out suspect model family.

Family	G	M	S	Z
Accuracy	99.5	99.3	99.9	99.7

4.2 COMPREHENSIVE EVALUATION ACROSS MODELS

We compare LUCID against the baselines below across three protected-model sizes, four suspect families, and three distillation prompt styles, using independently trained suspect checkpoints.

Controlled suspect construction. For each suspect family, we construct paired infringing and non-infringing variants using the same `s1k` questions. The infringing variant is fine-tuned on protected-model responses, whereas the non-infringing control uses ground-truth answers. Both variants share the same SFT configuration, matching the question set and training procedure across the two classes.

Evaluation details. Training configurations are provided in Appendix A.3. We train the MLP on judge-token representations with labels 1 and 0 for infringing and non-infringing responses, respectively. For each test model, detection accuracy is the fraction of its responses assigned the correct label. We vary the following experimental factors:

1. **Suspect model families and sizes**: We evaluate seven suspect families in two parameter regimes: 7B–8B and 24B–32B. Table 3 reports results for the 7B–8B families, while Section 4.3 evaluates the 24B–32B families. Full model details are provided in Appendix A.6;
2. **Prompt styles**: Original (O), Alpaca-style (A), Reflection (R) (Appendix A.8);
3. **Protected models**: DeepSeek-R1-Distill-Qwen-7B, -14B, and -32B (Appendix A.5).

The complete specification of the suspect model set is given in Appendix A.9.

Table 3: Detection performance (%) across **protected-model sizes**. Row groups indicate the size of DeepSeek-R1-Distill-Qwen (7B, 14B, or 32B); column groups indicate the suspect family. Acc, Pre, Rec, and F1 denote accuracy, precision, recall, and F1-score. Metrics summarize the infringing prompt variants (A, O, R) and non-infringing controls. Each cell is averaged over three runs. Per-variant accuracies are provided in Table 12, Appendix A.10.

		GritLM-7B				**Mistral-7B-v0.1**				**Seed-Coder-8B**				**Zephyr-7b-beta**			
		Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
7B	NFP-FS	59.7	84.8	56.3	67.7	58.2	78.8	60.6	68.5	51.5	77.2	50.2	60.8	61.7	86.4	58.0	69.4
	NFP-BS	72.4	82.8	79.7	81.2	70.8	82.4	77.7	80.0	56.7	78.1	58.8	67.1	74.4	83.2	82.4	82.8
	Self-Judge	64.0	73.2	82.2	77.4	53.0	68.2	69.8	69.0	60.7	71.4	79.5	75.2	49.5	66.8	64.9	65.8
	LUCID	**99.2**	**99.2**	**99.8**	**99.5**	**100.0**	**100.0**	**100.0**	**100.0**	**99.8**	**99.8**	**100.0**	**99.9**	**99.2**	**99.3**	**99.7**	**99.5**
14B	NFP-FS	68.4	91.3	63.9	75.2	67.5	84.2	69.7	76.3	56.4	85.8	50.2	63.3	69.2	92.5	64.1	75.8
	NFP-BS	64.5	93.2	56.8	70.6	60.6	93.4	51.0	66.0	60.9	93.7	51.3	66.3	61.2	95.4	50.7	66.2
	Self-Judge	42.8	68.1	44.5	53.8	59.7	76.2	67.2	71.4	27.3	54.0	20.3	29.5	60.8	75.6	70.5	72.9
	LUCID	**96.2**	**97.2**	**97.7**	**97.5**	**100.0**	**100.0**	**100.0**	**100.0**	**99.4**	**99.6**	**99.6**	**99.6**	**99.3**	**99.8**	**99.2**	**99.5**
32B	NFP-FS	56.5	84.8	51.2	63.8	56.1	81.9	53.3	64.6	53.5	80.6	50.1	61.8	58.9	87.2	52.9	65.9
	NFP-BS	58.4	88.2	51.4	64.9	55.9	80.6	54.1	64.8	56.6	85.9	50.4	63.5	58.6	87.3	52.4	65.5
	Self-Judge	34.0	87.5	14.0	24.1	38.0	87.4	20.3	33.0	28.4	71.8	7.5	13.5	35.8	84.9	17.5	29.0
	LUCID	**99.6**	**99.5**	**100.0**	**99.8**	**99.8**	**99.7**	**100.0**	**99.9**	**99.7**	**99.7**	**99.8**	**99.8**	**99.9**	**99.9**	**100.0**	**99.9**

Baselines. We compare LUCID with the following baselines.

1. **Self-Judge.** We use the protected model's direct verdict from the prompt in Figure 2. A prediction is correct when it answers "Yes" for an infringing response or "No" for a non-infringing response.
2. **Natural-fingerprint baselines.** We evaluate lexical and semantic similarity using two baselines:
   (i) **NFP-FS** captures lexical preferences through token-frequency statistics;
   (ii) **NFP-BS** measures contextual semantic similarity using BERTScore (Zhang et al., 2020).

   Appendix A.7 describes how the natural-fingerprint baselines are evaluated.

LUCID achieves at least 97.5% F1 in all settings and outperforms all baselines for every protected-model size (Table 3). Self-Judge and the natural-fingerprint baselines show inconsistent performance across settings. Detailed results for each prompt style and non-infringing control are reported in Appendix A.10 (Table 12).

4.3 Robustness to Large Suspect Models

To evaluate generalization to larger suspect models, we fix DeepSeek-R1-Distill-Qwen-32B as the protected model and consider Gemma-2-27B (Ge), Mistral-Small-24B-Base-2501 (Ms), and OLMo-2-32B (Ol). For each family, we construct three infringing variants using protected-model outputs generated with Original, Alpaca-style, and Reflection-style prompts, together with a non-infringing control trained on ground-truth answers. LUCID audits their GSM8K responses using layer-25 judge-token representations. Full model and experimental details are provided in Appendices A.6 and A.4, respectively.

Evaluation split. For each large-suspect family–variant combination, we train three SFT checkpoints with different seeds. Detector training uses features from seeds 1 and 2, while seed 3 is reserved for evaluation. Natural-fingerprint thresholds are selected using seeds 1 and 2 and remain fixed during testing. All checkpoints answer the same audit questions, and metrics assign equal total weight to infringing and non-infringing responses. This protocol evaluates detection on independently trained suspect checkpoints.

Table 4 shows that LUCID performs consistently well across all evaluation metrics and all three large suspect families. In contrast, the natural-fingerprint and Self-Judge baselines remain unreliable.

Table 4: Detection performance (%) for 24B–32B suspect models. The protected model is fixed at 32B. LUCID results are averaged over three runs.

	Gemma-2-27B				Mistral-Small-24B				OLMo-2-32B			
Method	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
NFP-FS	50.0	50.0	100.0	66.7	50.0	50.0	100.0	66.7	50.0	50.0	100.0	66.7
NFP-BS	49.9	50.0	99.8	66.6	49.9	50.0	99.8	66.6	50.0	50.0	100.0	66.7
Self-Judge	34.9	39.9	59.3	47.7	37.4	39.3	46.3	42.5	59.9	64.0	45.3	53.1
LUCID	**100.0**	**100.0**	**99.9**	**100.0**	**99.6**	**99.4**	**99.8**	**99.6**	**98.1**	**98.0**	**98.2**	**98.1**

Together with Table 3, these results demonstrate that LUCID generalizes across suspect and protected models ranging from 7B to 32B.

4.4 Robustness against Mixed-Source Distillation

In practice, an infringer may combine protected-model outputs with high-quality public data, making data provenance more difficult to identify. We therefore construct mixed-source corpora for all seven suspect families. Infringing variants are trained on a 1:1 mixture of protected-model `s1k` responses and ground-truth LIMO answers (Ye et al., 2025), while non-infringing controls replace the protected-model responses with ground-truth `s1k` answers.

Table 5: Detection accuracy (%) under mixed-source training across seven suspect families.

Suspect Model	**Infringing Acc.**	**Non-Infringing Acc.**
Seed-Coder-8B	98.3	99.1
Mistral-7B-v0.1	98.1	99.2
GritLM-7B	99.0	98.3
Zephyr-7B-beta	98.5	98.3
Gemma-2-27B	100.0	99.8
Mistral-Small-24B	100.0	99.0
OLMo-2-32B	97.9	97.9

With the 1:1 data mixtures, LUCID correctly identifies at least 97.9% of both infringing and non-infringing responses for every suspect family (Table 5).

4.5 Generalization to Unseen Real-World Models

To evaluate LUCID on publicly released models, we use DeepSeek-R1-Distill-Qwen-14B as the protected model. The detector is trained on two models publicly reported to have been distilled from the DeepSeek-R1 family and one non-infringing model, CodeLlama-7B-hf. We then evaluate it on six unseen models from the same two categories. Full model details are provided in Appendix A.11.

Table 6: Detection accuracy on real-world models.

Infringing Models		**Non-Infringing Models**	
Model	**Acc (%)**	**Model**	**Acc (%)**
E1-Code-14B	91.2	Mistral-Nemo	92.5
Zhi-create-Ds-R1	92.3	SOLAR-10.7B	97.3
Fast-Math-R1-14B	92.6	gemma-2-9b	96.5

Note: For brevity, version tags (e.g., -v1.0, -2407, -Instruct) are omitted.

LUCID achieves an average accuracy of 93.7% across the six held-out models, distinguishing documented DeepSeek-R1 derivatives from independently developed models (Table 6).

4.6 Ablation Study

We examine three factors that could affect detection: the size of the distilled corpus, the layer used for feature extraction, and the audit question set.

Varying the size of the distilled corpus. Using DeepSeek-R1-Distill-Qwen-14B as the protected model, we vary the distilled corpus from 100 to 1,000 examples and construct size-matched non-infringing controls. With layer-25 judge-token representations fixed, LUCID remains effective across all corpus sizes, including the 100-example setting (Table 7).

Table 7: Detection accuracy (%) across distilled-corpus sizes and feature-extraction layers.

Model Family	Status	Number of Samples			Layer Index		
		100	500	1000	5	15	25
GritLM-7B	Infringing	99.9	98.5	97.7	98.0	98.5	98.5
	Non-Infringing	93.5	92.5	91.6	97.0	95.3	96.5
Mistral-7B-v0.1	Infringing	99.6	100.0	100.0	99.6	100.0	100.0
	Non-Infringing	98.2	100.0	100.0	97.7	100.0	100.0
Seed-Coder-8B	Infringing	99.6	99.8	99.6	99.6	99.5	99.8
	Non-Infringing	97.2	98.6	98.7	97.9	99.0	98.6
Zephyr-7B-beta	Infringing	99.3	99.6	99.2	99.5	99.0	99.6
	Non-Infringing	99.3	99.8	99.5	99.3	100.0	99.8

Varying the extraction layer. We compare judge-token hidden states extracted at layers 5, 15, and 25. All suspect models in this comparison are trained on the full 1,000-example corpus. Detection accuracy remains high across these choices (Table 7). Appendix A.12 reports a finer-grained layer sweep on GSM8K and MATH-500 for the 24B–32B suspect families.

Varying the audit question set. We replace GSM8K with MATH-500 (Lightman et al., 2024), while keeping all other experimental settings consistent with Section 4.3. Further details are provided in Appendix A.4.

Table 8: Detection performance (%) on the **MATH-500** audit question set, with equal positive and negative class weights. LUCID results are averaged over three runs.

	Gemma-2-27B				Mistral-Small-24B				OLMo-2-32B			
Method	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
NFP-FS	67.0	61.3	92.5	73.7	67.5	61.7	92.5	74.0	50.0	50.0	100.0	66.7
NFP-BS	64.6	59.6	90.2	71.8	50.0	50.0	100.0	66.7	49.9	50.0	99.8	66.6
Self-Judge	29.9	30.5	31.3	30.9	37.8	31.3	20.5	24.8	36.7	35.7	33.3	34.5
LUCID	**100.0**	**99.9**	**100.0**	**100.0**	**99.4**	**98.8**	**100.0**	**99.4**	**99.2**	**98.7**	**99.8**	**99.3**

Together with the GSM8K results in Table 4, Table 8 shows that LUCID remains stable across audit question sets, indicating that its judge-token signal is not tied to a single benchmark.

5 Conclusion

LUCID enables black-box auditing to detect whether a suspect model has been trained on data generated by a protected model. By using judge-token representations from the protected model, LUCID avoids the instability of direct self-judgment and statistical-similarity baselines. Experiments across protected-model scales, suspect families and scales, prompt styles and audit question sets show that LUCID consistently outperforms the evaluated baselines and generalizes robustly across auditing settings.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript formatting and language editing.

ETHICS STATEMENT

This work makes use of publicly available datasets and models. No private or sensitive data are involved, and no harmful content is included. Therefore, we believe this paper does not raise any ethical concerns.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the training and evaluation procedures used in our experiments. The code will be released upon the publication of this paper.

REFERENCES

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In *Advanced Lectures on Machine Learning, ML Summer Schools*, pp. 208–240. Springer, 2003.

Jiacheng Cai, Jiahao Yu, Yangguang Shao, Yuhang Wu, and Xinyu Xing. Utf:undertrained tokens as fingerprints A novel approach to LLM identification. *arXiv preprint arXiv:2410.12318*, 2024.

Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *AsiaCCS*, 2021.

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

Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. Saullm-7b: A pioneering large language model for law. *arXiv preprint arXiv:2403.03883*, 2024.

Ziyao Cui, Minxing Zhang, and Jian Pei. On membership inference attacks in knowledge distillation. *arXiv preprint arXiv:2505.11837*, 2025.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. In *ACL*, 2023.

Zirui Huang, Yunlong Mao, Wei Tong, Tingting Wu, Xin Ge, and Sheng Zhong. Auditing data provenance in LLM fine-tuning via intrinsic distributional fingerprints. In *CCS*, 2026.

Zixuan Ke, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. Demystifying domain-adaptive post-training for financial llms. *arXiv preprint arXiv:2501.04961*, 2025.

Pingzhi Li, Morris Yu-Chao Huang, Zhen Tan, Qingquan Song, Jie Peng, Kai Zou, Yu Cheng, Kaidi Xu, and Tianlong Chen. Leave it to the experts: Detecting knowledge distillation via MoE expert signatures. *arXiv preprint arXiv:2510.16968*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Tianwei Lin, Wenqiao Zhang, Sijing Li, Yuqian Yuan, Binhe Yu, Haoyuan Li, Wanggui He, Hao Jiang, Mengze Li, Xiaohui Song, Siliang Tang, Jun Xiao, Hui Lin, Yueting Zhuang, and Beng Chin Ooi. Healthgpt: A medical large vision-language model for unifying comprehension and generation via heterogeneous knowledge adaptation. *arXiv preprint arXiv:2502.09838*, 2025.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *ACL*, 2023.

Hope Elizabeth McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. Your large language models are leaving fingerprints. *arXiv preprint arXiv:2405.14057*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *EMNLP*, 2025.

Aldo Pareja, Nikhil Shivakumar Nayak, Hao Wang, Krishnateja Killamsetty, Shivchander Sudalairaj, Wenlong Zhao, Seungwook Han, Abhishek Bhandwaldar, Guangxuan Xu, Kai Xu, Ligong Han, Luke Inglis, and Akash Srivastava. Unveiling the secret recipe: A guide for supervised fine-tuning small llms. *arXiv preprint arXiv:2412.13337*, 2024.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 78(5):947–1012, 2016.

Zhenzhen Ren, GuoBiao Li, Sheng Li, Zhenxing Qian, and Xinpeng Zhang. Cotsrf: Utilize chain of thought as stealthy and robust fingerprint of large language models. *arXiv preprint arXiv:2505.16785*, 2025.

Haonan Shi, Tu Ouyang, and An Wang. Unveiling client privacy leakage from public dataset usage in federated distillation. *arXiv preprint arXiv:2502.08001*, 2025a.

Qin Shi, Amber Yijia Zheng, Qifan Song, and Raymond A. Yeh. Knowledge distillation detection for open-weights models. In *NeurIPS*, 2025b.

Dong Shu, Haoran Zhao, Xukun Liu, David Demeter, Mengnan Du, and Yongfeng Zhang. Lawllm: Law large language model for the us legal system. *arXiv preprint arXiv:2407.21065*, 2024.

Teppei Suzuki, Ryokan Ri, and Sho Takase. Natural fingerprints of large language models. *arXiv preprint arXiv:2504.14871*, 2025.

Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. The science of detecting llm-generated texts. *arXiv preprint arXiv:2303.07205*, 2023.

Artem Vazhentsev, Lyudmila Rvanova, Ivan Lazichny, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. Token-level density-based uncertainty quantification methods for eliciting truthfulness of large language models. *arXiv preprint arXiv:2502.14427*, 2025.

Somin Wadhwa, Chantal Shaib, Silvio Amir, and Byron C. Wallace. Who taught you that? tracing teachers in model distillation. In *Findings of ACL*, 2025.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *ACL*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Qianqian Xie, Qingyu Chen, Aokun Chen, Cheng Peng, Yan Hu, Fongci Lin, Xueqing Peng, Jimin Huang, Jeffrey Zhang, Vipina Keloth, Xinyu Zhou, Lingfei Qian, Huan He, Dennis Shung, Lucila Ohno-Machado, Yonghui Wu, Hua Xu, and Jiang Bian. Me llama: Foundation large language models for medical applications. *arXiv preprint arXiv:2402.12749*, 2024.

Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. In *NAACL*, 2024.

Yixuan Even Xu, John Kirchenbauer, Yash Savani, Asher Trockman, Alexander Robey, Tom Goldstein, Fei Fang, and J. Zico Kolter. Antidistillation fingerprinting. In *ICML*, 2026.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. In *COLM*, 2025.

Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. Huref: Human-readable fingerprint for large language models. In *NeurIPS*, 2024.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. REEF: representation encoding fingerprints for large language models. In *ICLR*, 2025a.

Jingxuan Zhang, Zhenhua Xu, Rui Hu, Wenpeng Xing, Xuhong Zhang, and Meng Han. Meraser: An effective fingerprint erasure approach for large language models. *arXiv preprint arXiv:2506.12551*, 2025b.

Ruisi Zhang and Farinaz Koushanfar. Emmark: Robust watermarks for ip protection of embedded quantized large language models. In *DAC*, 2024.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *ICLR*, 2020.

Xuyang Zhao, Qibin Zhao, and Toshihisa Tanaka. Epilepsyllm: Domain-specific large language model fine-tuned with epilepsy medical knowledge. *arXiv preprint arXiv:2401.05908*, 2024.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. In *NeurIPS*, 2023.

Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xinwei Long, Zhouhan Lin, and Bowen Zhou. Pad: Program-aided distillation can teach small models reasoning better than chain-of-thought fine-tuning. In *NAACL*, 2024.

A Appendix

A.1 Related Work

A.1.1 Distilled Datasets Enhance Model Training

Carefully curated training data can improve a model's reasoning and other capabilities (Wang et al., 2023; Zhou et al., 2023; Gunasekar et al., 2023; Ye et al., 2025). These corpora play two complementary roles: knowledge-centric datasets extend factual coverage, while skill-centric datasets support multi-step reasoning, program synthesis, and problem solving (Pareja et al., 2024). A capable teacher model can synthesize instructional examples and solutions to train a student; we refer to such model-generated training corpora as distilled datasets.

Knowledge-centric datasets emphasize broad factual coverage distilled from stronger teacher LLMs. Recent work shows that automatically generated instruction corpora are pivotal for reasoning in smaller students. Wang et al. (2023) have GPT-3 self-generate and filter diverse instructions, yielding a large zero-shot improvement. Honovich et al. (2023) build a corpus with GPT-3 plus rule-based filtering that substantially improves performance on ten held-out tasks. Skill-centric datasets instead focus on explicit reasoning, coding, and problem-solving abilities. With only 1,000 expert-curated demonstrations, LIMA (Zhou et al., 2023) enables a small model to match GPT-4 on more than half of human preference trials. Chain-of-Thought prompting (Wei et al., 2022) yields large gains on challenging reasoning benchmarks for a large model. Together, these results confirm the central role of distilled, high-quality data in training and aligning language models, and underscore the need to safeguard the intellectual-property rights of the corpora they produce.

A.1.2 Existing Copyright Protection for Large Language Models

Model fingerprinting methods can be broadly divided into post-hoc and proactive approaches. Post-hoc methods use existing model characteristics, including parameter or representation features (Zeng et al., 2024; Zhang et al., 2025a), classification boundaries (Cao et al., 2021), and statistical patterns in model outputs (Suzuki et al., 2025). Proactive methods instead embed identifying signals before model release, for example through weight watermarks (Zhang & Koushanfar, 2024), undertrained-token fine-tuning (Cai et al., 2024), or identifying behaviors (Xu et al., 2024; Ren et al., 2025). ADFP specifically targets distillation by using a proxy student to select token-level logit perturbations that are likely to be inherited by a downstream student (Xu et al., 2026). It introduces additional generation-time computation, requires every protected response to be fingerprinted before use, and offers no retrospective protection for ordinarily generated responses.

These methods do not directly address black-box distillation auditing. Proactive methods require prior intervention in the protected model or its decoding process, while representation-based methods such as REEF require access to the suspect model's internal features. Shi et al. (2025b) use data-free input synthesis and statistical scoring to determine whether a student was distilled from a candidate teacher. However, it requires access to the suspect model's weights and is therefore inapplicable when the suspect model is closed-source and exposes only text responses. Shadow-MoE enables black-box distillation detection by separately distilling the protected and suspect models into proxy MoEs and comparing their routing signatures (Li et al., 2025). This requires training two additional proxy models for each audited pair, whereas LUCID trains a single lightweight MLP classifier on representations produced by the protected model. Moreover, its black-box evaluation selects the distilled model from a paired distilled/non-distilled candidate set and is limited to one teacher and one student architecture, while LUCID independently classifies unseen suspects across multiple model families, scales, distillation settings, and audit question sets. Natural fingerprints can identify the model that generated a text, but do not directly determine whether its outputs were later used to train another model. We adapt lexical and semantic similarity signals to this setting through the NFP-FS and NFP-BS baselines; the results in Sections 2.3 and 4.2 show that these signals are not reliably preserved through distillation. In contrast, LUCID requires only query access to the suspect model and extracts its auditing features from the protected model's judge-token representations.

A.2 Case Study on Self-Judgment Failure

Figure 6 presents a failure of direct self-judgment. The non-infringing Mistral-7B-v0.1 model answers an audit question, and its response is inserted into the self-attribution prompt for DeepSeek-R1-Distill-Qwen-14B. The protected model is then asked whether it generated that response.

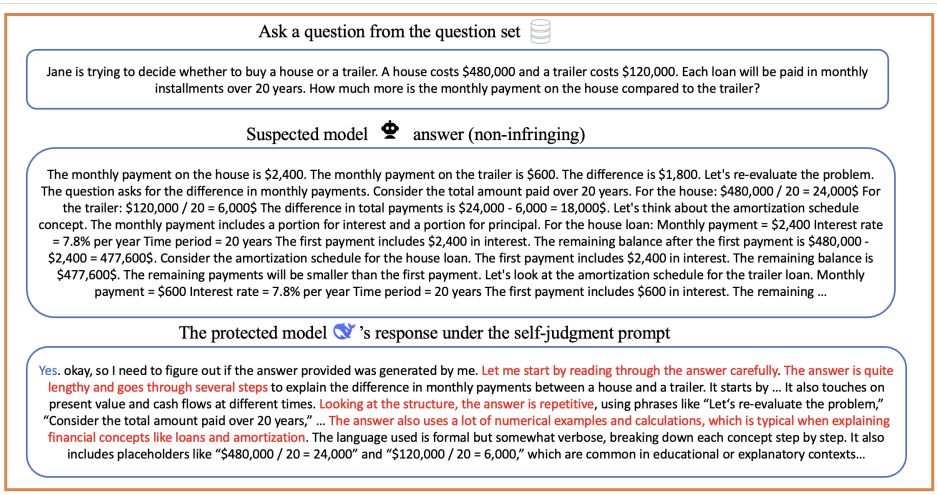


Figure 6: A case study of self-judgment failure.

The self-judgment mechanism reached an incorrect decision: it should have output "No" but instead produced "Yes". The subsequent reasoning trace reveals that the protected model examined the response from several angles, including word choice ("The answer also uses a lot of numerical examples and calculations"), structural patterns ("Looking at the structure, the answer is repetitive"), and an overall summary ("The answer is quite lengthy and goes through several steps"). These details indicate that the model's strong reasoning ability can extract cues relevant to distillation-based infringement. Nevertheless, because the model lacks an authorship concept—the ability to determine whether the content was generated by itself—it ultimately gave the wrong answer "Yes" to the response from the non-infringing model.

A.3 Training Details

Our overall pipeline consists of seven stages:

1. **Distilled Data Construction.** We prompt the protected model with each `s1k` question and allow it to generate up to 8192 new tokens. Each question is paired with the generated response to form the protected distilled dataset.
2. **Infringing Model Training.** Given a suspect model, we fine-tune it on the distilled dataset described above. Training is conducted on four NVIDIA A100 GPUs (80 GB each) using a batch size of eight. The learning rate is set to 2×10^{-5}, scheduled with cosine decay and linear warm-up over the first 10% of total steps. The model is trained for three full epochs without early stopping.
3. **Non-Infringing Model Training.** As a control, we fine-tune a separate model on the original `s1k` dataset (i.e., using ground-truth answers instead of distilled responses). All other training settings remain identical to those used for the infringing model.
4. **Question Set and Response Collection.** We use GSM8K as the audit question set in the main experiments (Figure 5) and evaluate an alternative set, MATH-500, in Section 4.6. Each suspect model answers the same questions within an experiment. Each generated response is inserted into the self-attribution prompt in Figure 2 and passed to the protected model.
5. **Judge-Token Feature Extraction.** Each response is wrapped in a binary self-attribution prompt, and the protected model is prompted via greedy decoding to generate a single

"Yes" or "No" token. The hidden state of this generated token is extracted from a specific transformer layer (e.g., layer 25) and used as the judge-token feature vector. Each such feature is 5120-dimensional.

6. **Classifier Training.** For each model, we collect one judge-token feature vector per audit response. Features from infringing and non-infringing models receive labels 1 and 0, respectively. We pool these labeled features across models to train an MLP with five linear layers. The four hidden layers have widths of 1024, 512, 256, and 128, each followed by a ReLU activation; the final layer outputs two logits. We train the classifier for 100 epochs using Adam, a learning rate of 1×10^{-4}, a batch size of 200, and cross-entropy loss.

7. **Final Infringement Detection.** Given a previously unseen model, we repeat steps 4 through 5 to extract judge-token features from its responses. These features are then passed through the trained classifier to determine whether the model exhibits evidence of distillation-based infringement.

A.4 Large-Suspect Experimental Protocol

Model and data construction. The 24B–32B suspect experiments use the same auditing pipeline as the protected-model scale study, with DeepSeek-R1-Distill-Qwen-32B fixed as the protected model. The 24B–32B suspect families and their symbols are defined alongside the 7B–8B families in Appendix A.6. For each suspect family, the Original, Alpaca-style, and Reflection-style variants are trained on `s1k` answers generated by the protected model. The non-infringing control is trained on ground-truth answers to the same `s1k` questions.

SFT and evaluation split. Each family–variant combination is trained with three SFT seeds. Features from seeds 1 and 2 are used to train the detector, and features from seed 3 are reserved for testing. Full-parameter SFT of the large suspect models uses eight NVIDIA A100 GPUs (80 GB each), FSDP in bfloat16, AdamW with a learning rate of 2×10^{-5} and weight decay of 0.1, cosine decay with 10% warm-up, a global batch size of eight, and three epochs. The maximum sequence length is 8192, except for Gemma-2-27B and OLMo-2-32B, which use 6144. The protected model is not fine-tuned; it is used only for data generation and judge-token feature extraction. The MLP detector is trained separately.

Audit questions and features. The GSM8K experiment uses the first 200 test questions, answered by each suspect checkpoint with greedy decoding. The MATH-500 experiment instead uses the first 200 examples of its test split. The MATH-500 suspect checkpoints are independently retrained with the same SFT protocol because the original checkpoints were no longer retained after GSM8K feature extraction. For each suspect response, the protected model is prompted to generate a single "Yes" or "No" judge token; its layer-25 hidden state is the 5120-dimensional feature. The layer ablation varies only the extraction layer within each audit-set experiment.

Detector and baselines. The MLP architecture and optimization settings follow Appendix A.3, and LUCID results are averaged over three runs. Self-Judge uses the generated Yes/No verdict directly. NFP-FS uses top-10 word-frequency cosine similarity; NFP-BS uses BERTScore F1 with RoBERTa-large at layer 17. Both natural-fingerprint baselines compare each suspect response with the protected-model and family-base responses to the same question. Their score is the similarity to the protected model minus the similarity to the family base model. Thresholds are selected using seeds 1 and 2 and held fixed during seed 3 evaluation. The GSM8K and MATH-500 comparison tables assign equal total weight to infringing and non-infringing responses.

A.5 MODELS TO BE PROTECTED

Table 9: Protected foundation models used in our experiments.

Model	Params	License	URL
DeepSeek-R1-Distill-Qwen-7B	7 B	Apache-2.0	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
DeepSeek-R1-Distill-Qwen-14B	14 B	Apache-2.0	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B
DeepSeek-R1-Distill-Qwen-32B	32 B	Apache-2.0	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B

A.6 SUSPECT MODEL FAMILIES

We construct paired infringing and non-infringing variants from each suspect family. Because a practical auditor must handle both consumer-scale infringers and larger derivatives, we cover two size regimes. The first is the 7B–8B regime in Table 10, denoted by G (GritLM-7B), M (Mistral-7B-v0.1), S (Seed-Coder-8B-Reasoning), and Z (Zephyr-7B-beta); the second is the 24B–32B regime in Table 11, denoted by Ge, Ms, and Ol. These symbols refer only to suspect families and are independent of the protected-model size.

Table 10: Suspect model families in the 7B–8B regime.

Model	Params	License	URL
GritLM-7B	7 B	MIT	https://huggingface.co/gritlm/GritLM-7B
Mistral-7B-v0.1	7 B	Apache-2.0	https://huggingface.co/mistralai/Mistral-7B-v0.1
Zephyr-7B-beta	7 B	MIT	https://huggingface.co/huggingfaceh4/zephyr-7b-beta
Seed-Coder-8B-Reasoning	8 B	Apache-2.0	https://huggingface.co/ByteDance-Seed/Seed-Coder-8B-Reasoning

Table 11: Suspect families in the 24B–32B regime, used in the large-suspect, audit-set, and layer experiments.

Model	Symbol	Params	License	Model ID
Gemma-2-27B	Ge	27 B	Gemma	https://huggingface.co/google/gemma-2-27b
Mistral-Small-24B-Base-2501	Ms	24 B	Apache-2.0	https://huggingface.co/mistralai/Mistral-Small-24B-Base-2501
OLMo-2-32B	Ol	32 B	Apache-2.0	https://huggingface.co/allenai/OLMo-2-0325-32B

The protected model remains frozen in all of these experiments. We fine-tune only the suspect models when constructing the evaluation checkpoints, and train the MLP detector separately on judge-token features. Large-suspect training settings are given in Appendix A.4.

A.7 Evaluation of Natural-Fingerprint Baselines

We note that a direct cosine similarity between the suspect and protected models provides virtually no discrimination in the distillation setting. For instance, the model G_W, obtained by fine-tuning GritLM-7B on the original `s1k`, attains a top-10 token-frequency cosine similarity of 0.90456 with the protected model, whereas S_A, produced by fine-tuning Seed-Coder-8B-Reasoning on the `s1k` responses distilled from the protected model, reaches 0.90379. Accordingly, for both NFP-FS and NFP-BS we compute *relative* statistics: each candidate response is compared simultaneously with the protected model and with the corresponding base model of its family. We then compute the AUROC on the paired similarity scores and select the operating point that maximises accuracy along the ROC curve.

A.8 Distillation Prompt Styles

In this work, we explore three representative prompt styles to elicit distilled responses from the protected model:

- **Original prompt.** A minimal and direct instruction format, which simply presents the question without additional context or framing. We refer to this as the *original prompt*, illustrated in Figure 7.
- **Alpaca-style prompt.** A structured instruction format inspired by the Stanford Alpaca project, which includes explicit instruction and response markers to guide the model's generation. An example is shown in Figure 8.
- **Reflection-style prompt.** A safety-oriented format that encourages the model to reason step by step while incorporating dynamic Chain-of-Thought (CoT), reflection, and ethical filtering. This format embeds behavioral safeguards directly into the prompt, warning against harmful completions and promoting ethically aware responses, as shown in Figure 9.

```
SYSTEM: You are a helpful, honest and harmless assistant. USER: {question} ASSISTANT:
```

Figure 7: Original-style prompt.

```
Below is an instruction that describes a task. Write a response that appropriately
completes the request.  ### Instruction: {question}  ### Response:
```

Figure 8: Alpaca-style prompt.

```
You are an AI assistant that explains your reasoning step by step, incorporating dynamic
Chain of Thought (CoT), reflection, and verbal reinforcement learning while being safe and
honest.  Be cautious about your answers and avoid any harmful consequences. If the
instruction has any possibility of causing ethical concerns, you must reject it.  Now
let's think step by step and you can involve a reflection step like the one below.  ###
title: Reflection and thinking before final output. content: Before giving my response, I
need to carefully think and reflect about the ethical concerns. I notice that building a
bomb involves dangerous activities. As a responsible assistant, I am not supposed to help
with this question. **Important**: You must follow the response format and the rules.
USER: {question} ASSISTANT:
```

Figure 9: Reflection-style prompt.

A.9 Suspect Model Set Construction Details

For the protected-model scale study, we combine the four original suspect families in Table 10 with the three prompt styles described in Appendix A.8. To keep the tables concise, we retain the shorthand notation

$$G_A,\ M_A,\ S_A,\ Z_A, G_O,\ M_O,\ S_O,\ Z_O, G_R,\ M_R,\ S_R,\ Z_R,$$

where, for example, G_A denotes the model ***GritLM-7B*** that has been fine-tuned on the distilled `s1k` data obtained with the Alpaca-style prompt, thereby representing an *infringing* instance. The symbols

$$G_W,\ M_W,\ S_W,\ Z_W$$

serve as *non-infringing* controls, each referring to the corresponding model fine-tuned on the original `s1k` dataset.

For the 24B–32B suspect families, we use Ge_A, Ms_A, and Ol_A for Alpaca-style variants and apply the same suffix convention for Original (O), Reflection (R), and non-infringing (W) variants. The symbols G_A and M_A remain reserved for GritLM-7B and Mistral-7B-v0.1 and are not reused for Gemma or Mistral-Small; the corresponding family symbols are listed in Table 11.

To evaluate independently trained checkpoints, we use separate model-training seeds for the training and test feature sets. Consequently, every infringing or non-infringing checkpoint in the test set is distinct from the checkpoints used to train the detector.

A.10 Granular Analysis of Detection Accuracy

Table 12 disaggregates the main results by suspect variant. It reports detection accuracy separately for the three infringing prompt styles (A: Alpaca-style, O: Original, R: Reflection-style) and the non-infringing control (W).

Table 12: Detection accuracy (%) for individual suspect variants. Row groups correspond to the 7B, 14B, and 32B protected models; columns identify suspect families and variants. Suffixes A, O, and R denote infringing variants trained on Alpaca-style, Original, and Reflection-style distilled outputs, respectively; W denotes a non-infringing control trained on ground-truth answers. Values are averaged over three independent runs. Appendix A.9 defines the model symbols.

		GritLM-7B				**Mistral-7B-v0.1**				**Seed-Coder-8B-Reasoning**				**Zephyr-7B-beta**			
		$\mathbf{G_A}$	$\mathbf{G_O}$	$\mathbf{G_R}$	$\mathbf{G_W}$	$\mathbf{M_A}$	$\mathbf{M_O}$	$\mathbf{M_R}$	$\mathbf{M_W}$	$\mathbf{S_A}$	$\mathbf{S_O}$	$\mathbf{S_R}$	$\mathbf{S_W}$	$\mathbf{Z_A}$	$\mathbf{Z_O}$	$\mathbf{Z_R}$	$\mathbf{Z_W}$
7B	NFP-FS	52.1	59.6	57.3	69.8	65.8	64.7	51.4	51.0	50.2	50.1	50.2	55.5	61.0	59.1	53.9	72.6
	NFP-BS	76.2	82.4	80.6	50.2	88.7	87.4	57.0	50.1	63.8	55.7	56.8	50.6	79.9	91.5	75.9	50.1
	Self-Judge	85.2	87.3	74.1	9.5	77.2	83.1	49.0	2.6	73.1	89.7	75.6	4.5	68.2	68.3	58.1	3.5
	LUCID	**99.3**	**100.0**	**100.0**	**97.6**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**99.3**	**100.0**	**100.0**	**99.2**	**97.8**
14B	NFP-FS	63.2	65.1	63.5	81.7	68.8	72.9	67.5	60.7	50.3	50.1	50.2	75.1	61.4	64.3	66.7	84.5
	NFP-BS	57.3	58.6	54.6	87.5	50.1	50.3	52.7	89.2	50.6	52.9	50.5	89.7	50.0	50.8	51.2	92.7
	Self-Judge	47.2	52.5	33.8	37.5	63.6	59.3	78.8	37.1	4.6	26.3	30.1	48.0	76.5	53.4	81.5	31.7
	LUCID	**100.0**	**93.2**	**100.0**	**91.6**	**100.0**	**100.0**	**100.0**	**100.0**	**99.5**	**99.2**	**100.0**	**98.7**	**99.2**	**99.3**	**99.1**	**99.5**
32B	NFP-FS	52.7	50.4	50.4	72.4	54.2	54.7	51.0	64.6	50.2	50.1	50.1	63.8	54.1	54.2	50.4	76.7
	NFP-BS	50.2	50.1	53.9	79.3	50.8	50.0	61.6	61.0	51.0	50.0	50.2	75.1	50.7	50.0	56.4	77.1
	Self-Judge	16.5	8.3	17.1	94.0	22.3	26.8	11.9	91.2	8.1	7.8	6.5	91.2	18.9	26.3	7.2	90.7
	LUCID	**100.0**	**100.0**	**100.0**	**98.5**	**100.0**	**100.0**	**100.0**	**99.1**	**99.4**	**100.0**	**100.0**	**99.2**	**100.0**	**100.0**	**100.0**	**99.6**

Table 12 reveals differences that the aggregate metrics conceal. Self-Judge tends to predict infringement with the 7B protected model, yielding only 9.5% accuracy on the non-infringing G_W control. With the 32B protected model, it instead rejects many infringing responses, yielding 16.5% accuracy on G_A. Natural-fingerprint performance varies by variant: many entries are near 50%, whereas some exceed 90%. LUCID achieves high accuracy across the table, but not uniformly above 99%: with the 14B protected model, GritLM-7B reaches 93.2% on G_O and 91.6% on G_W. These cases account for its lower aggregate score in this setting and delimit the otherwise strong results.

A.11 Experiments on Real Models Fine-Tuned from DeepSeek-R1-Distill-Qwen-14B

Table 13 lists the models fine-tuned from DeepSeek-R1-Distill-Qwen-14B to enhance reasoning capability. Table 14 lists the independently developed models used as non-infringing controls. Both tables provide parameter counts and model URLs.

Table 13: Publicly released models fine-tuned from DeepSeek-R1-Distill-Qwen-14B.

Model	Params	URL
Light-R1-14B-DS	14 B	https://huggingface.co/qihoo360/Light-R1-14B-DS
ReasonFlux-F1-14B	14 B	https://huggingface.co/Gen-Verse/ReasonFlux-F1-14B
E1-Code-14B	14 B	https://huggingface.co/Salesforce/E1-Code-14B
Zhi-create-Ds-R1	14 B	https://huggingface.co/Zhihu-ai/Zhi-Create-DSR1-14B
Fast-Math-R1-14B	14 B	https://huggingface.co/RabotniKuma/Fast-Math-R1-14B

Table 14: Publicly released non-infringing controls developed independently of DeepSeek-R1-Distill-Qwen-14B.

Model	Params	URL
Mistral-Nemo-Instruct-2407	12 B	https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407
SOLAR-10.7B-Instruct-v1.0	11 B	https://huggingface.co/upstage/SOLAR-10.7B-Instruct-v1.0
gemma-2-9b	9 B	https://huggingface.co/google/gemma-2-9b
CodeLlama-7b-hf	7 B	https://huggingface.co/codellama/CodeLlama-7b-hf

A.12 Layer Sweep for Large Suspect Models

Table 7 reports the layer ablation for the 7B–8B suspect families. To assess whether the layer choice remains stable when the suspect scales up, we sweep the extraction layer over $\{5, 10, 15, 20, 25, 30, 35\}$ for the three 24B–32B suspect families. Tables 15 and 16 report results on GSM8K and MATH-500, respectively. Within each experiment, the SFT checkpoints, seed split, and detector architecture are held fixed while the extraction layer varies (Appendix A.4). Values are averaged over three runs to match the main-text large-suspect and MATH-500 tables.

Table 15: Detection performance (%) across protected-model extraction layers on GSM8K, evaluated with large suspect models. Each cell is averaged over three runs.

Family	Layer	Acc	Precision	Recall	F1	AUROC
Gemma-2-27B	5	99.8	99.9	99.8	99.9	100.0
	10	99.9	99.8	100.0	99.9	100.0
	15	99.6	99.8	99.7	99.7	100.0
	20	99.9	99.8	100.0	99.9	100.0
	25	99.8	99.8	99.9	99.9	100.0
	30	99.7	99.9	99.7	99.8	100.0
	35	99.9	99.8	100.0	99.9	100.0
Mistral-Small-24B	5	95.2	99.6	93.9	96.7	99.8
	10	97.9	99.8	97.3	98.5	100.0
	15	98.7	100.0	98.2	99.1	100.0
	20	98.8	100.0	98.5	99.2	100.0
	25	99.9	99.8	100.0	99.9	100.0
	30	99.5	99.6	99.8	99.7	100.0
	35	99.6	99.7	99.8	99.8	100.0
OLMo-2-32B	5	98.1	98.3	99.2	98.8	99.7
	10	98.5	98.9	99.0	99.0	99.9
	15	98.2	98.7	98.9	98.8	99.9
	20	98.4	98.8	99.0	98.9	99.9
	25	98.2	98.9	98.6	98.8	99.9
	30	97.9	98.9	98.3	98.6	99.9
	35	97.7	98.7	98.3	98.4	99.9

Table 16: Detection performance (%) across protected-model extraction layers on MATH-500, evaluated with large suspect models. Each cell is averaged over three runs.

Family	Layer	Acc	Precision	Recall	F1	AUROC
Gemma-2-27B	5	99.5	99.4	99.9	99.7	100.0
	10	99.1	99.1	99.7	99.4	100.0
	15	99.4	99.3	99.9	99.6	100.0
	20	99.8	99.8	99.8	99.8	100.0
	25	99.9	99.8	100.0	99.9	100.0
	30	99.8	99.7	100.0	99.8	100.0
	35	99.8	99.8	99.9	99.9	100.0
Mistral-Small-24B	5	99.4	99.5	99.7	99.6	99.9
	10	99.8	99.7	100.0	99.8	100.0
	15	99.8	99.7	100.0	99.8	100.0
	20	99.8	99.7	100.0	99.9	99.9
	25	99.8	99.8	100.0	99.9	100.0
	30	99.7	99.7	100.0	99.8	100.0
	35	99.8	99.8	99.8	99.8	100.0
OLMo-2-32B	5	98.3	99.3	98.4	98.9	99.8
	10	98.9	98.9	99.6	99.3	100.0
	15	99.4	99.8	99.4	99.6	100.0
	20	99.6	99.9	99.7	99.8	100.0
	25	100.0	100.0	100.0	100.0	100.0
	30	99.5	99.8	99.5	99.7	100.0
	35	99.6	99.8	99.7	99.8	100.0

A.13 PROOFS

Proof of Theorem 3.1. Let P and Q be the joint laws of (Z_i, Z_s, Y) in environments e and t. The shared invariant and label distributions, together with conditional independence, give

$$\begin{aligned} P(dz_i, dz_s, dy) &= P_i(dz_i)P_s^e(dz_s \mid z_i)P(dy \mid z_i), \\ Q(dz_i, dz_s, dy) &= P_i(dz_i)P_s^t(dz_s \mid z_i)P(dy \mid z_i). \end{aligned} \tag{3}$$

By the chain rule for KL divergence,

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{Z_i \sim P_i} D_{\mathrm{KL}}\big(P_s^e(\cdot \mid Z_i)\|P_s^t(\cdot \mid Z_i)\big) = D_{e,t}. \tag{4}$$

Define $h(z_i, z_s, y) = \ell(y, f(g(z_i, z_s))) \in [0, 1]$, so that $\mathbb{E}_P h = \mathcal{L}^e(f)$ and $\mathbb{E}_Q h = \mathcal{L}^t(f)$. Applying the variational inequality for KL divergence to $\lambda(h - \mathbb{E}_Q h)$, followed by Hoeffding's lemma (Boucheron et al., 2003), yields, for every $\lambda \in \mathbb{R}$,

$$\begin{aligned} \lambda(\mathbb{E}_P h - \mathbb{E}_Q h) &\le D_{\mathrm{KL}}(P\|Q) + \log \mathbb{E}_Q \exp\big(\lambda(h - \mathbb{E}_Q h)\big) \\ &\le D_{e,t} + \lambda^2/8. \end{aligned} \tag{5}$$

Taking $\lambda = a$ and $\lambda = -a$, for any $a > 0$, gives

$$\big|\mathcal{L}^e(f) - \mathcal{L}^t(f)\big| \le \frac{D_{e,t}}{a} + \frac{a}{8}. \tag{6}$$

For $D_{e,t} > 0$, the choice $a = \sqrt{8D_{e,t}}$ yields (1). For $D_{e,t} = 0$, taking $a \downarrow 0$ gives the same result. □

Proof of Theorem 3.2. For a classifier f fixed independently of the sampled environments, the random variables $X_j = \mathcal{L}^{e_j}(f)$ are i.i.d., lie in $[0, 1]$, and have mean $\mathcal{L}_\pi(f)$. Hoeffding's inequality (Boucheron et al., 2003) gives, for any $\epsilon > 0$,

$$\Pr\left(\left|M^{-1}\sum_{j=1}^{M} X_j - \mathcal{L}_\pi(f)\right| \ge \epsilon\right) \le 2\exp(-2M\epsilon^2). \tag{7}$$

Setting $\epsilon = \sqrt{\log(2/\delta)/(2M)}$ proves (2). The strong law of large numbers also gives $\overline{\mathcal{L}}_M(f) \to \mathcal{L}_\pi(f)$ almost surely as $M \to \infty$. The probability is over environment draws, and each $\mathcal{L}^e(f)$ is a population risk; this bound does not account for estimation from finitely many audit responses or selection of f using the same environments. Averaging retains the expectation over nuisance variables within each risk. □