Protecting Language Models Against Unauthorized Distillation through Trace Rewriting

Xinhang Ma, William Yeoh, Ning Zhang, Yevgeniy Vorobeychik
Washington University in St. Louis
{m.owen, wyeoh, zhang.ning, yvorobeychik}@wustl.edu

Abstract

Knowledge distillation is a widely adopted technique for transferring capabilities from LLMs to smaller, more efficient student models. However, unauthorized use of knowledge distillation takes unfair advantage of the considerable effort and cost put into developing frontier models. We investigate methods for modifying teacher-generated reasoning traces to achieve two objectives that deter unauthorized distillation: (1) *anti-distillation*, or degrading the training usefulness of query responses, and (2) *API watermarking*, which embeds verifiable signatures in student models. We introduce several approaches for dynamically rewriting a teacher's reasoning outputs while preserving answer correctness and semantic coherence. Two of these leverage the rewriting capabilities of LLMs, while others use gradient-based techniques. Our experiments show that a simple instruction-based rewriting approach achieves a strong anti-distillation effect while maintaining or even improving teacher performance. Furthermore, we show that our rewriting approach also enables embedding watermarks that can be reliably detected with essentially no false alarms. Our code is available at `https://github.com/xhOwenMa/trace-rewriting`.

1 Introduction

Knowledge distillation is a simple learning technique for transferring knowledge from one model (the *teacher*) to another (the *student*) (Hinton et al., 2015). In its simplest form, distillation proceeds by querying a teacher model $\mathcal{T}$ with inputs x to obtain responses $y = \mathcal{T}(x)$, then training a student model $\mathcal{S}$ on the resulting input-output pairs using supervised fine-tuning. Since its introduction, knowledge distillation has become a major workhorse in machine learning across a broad array of applications (Gou et al., 2021; Sanh et al., 2019), as it enables using large, complex, and expensive to execute teachers to train smaller and more efficient student models. These student models can then be deployed at lower cost and exhibit lower inference time, which can be a critical enabler in real-time applications.

However, the simplicity and effectiveness of knowledge distillation even with only a black-box query access to the teacher $\mathcal{T}$—for example, when $\mathcal{T}$ is proprietary—means that this technique can also be used for knowledge and capability "stealing". This issue is especially acute with frontier reasoning-capable LLMs, the design and training of which carry enormous effort and expense (Guo et al., 2025; Jaech et al., 2024). These models produce explicit reasoning traces—structured outputs that decompose problem-solving into intermediate steps before arriving at a final answer. These traces provide rich supervision signals that go beyond mere input-output pairs. As a result, practitioners increasingly seek to distill reasoning capabilities from these frontier models. Since distillation incurs a fraction of the cost of training an original teacher, its efficacy without any guardrails can disincentivize innovation.

Two classes of approaches have been proposed to counter unauthorized distillation of large language models: *anti-distillation* and *API watermarking*. Anti-distillation methods aim to degrade the efficacy of distillation training (Li et al., 2025; Savani et al., 2025). However, state-of-the-art approaches for anti-distillation significantly degrade the *teacher* efficacy as well as the student, making them impractical. API watermarking, on the other hand, attempts to insert a watermark into the query responses in a way that enables its verification from the student model trained on such traces (He et al., 2022a,b; Zhao et al., 2022, 2023b). However, most API watermarking approaches use token-level statistics, and tend to result in a non-negligible false-alarm rate which provides plausible deniability for the model "thief".

We propose and analyze several methods for

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11307–11324

modifying teacher-generated reasoning traces to achieve these two complementary objectives. Specifically, we propose two classes of approaches for anti-distillation. The first uses an assistant LLM to rewrite instructions specifically in order to subvert their use in downstream training while preserving semantics. This approach leverages the semantic understanding of modern LLMs to transform clean traces into modified versions that achieve our objectives. Additionally, we adapt our instruction rewriting approach to facilitate stealthy watermark embedding in reasoning traces. Our second approach uses (projected) gradient-based optimization in the embedding space with the explicit objective of degrading training efficacy of a (proxy) student.

Conceptually, the problem of anti-distillation is closely related to the extensive literature on poisoning attacks in machine learning (Goldblum et al., 2022; Tian et al., 2022; Vorobeychik and Kantarcioglu, 2018), and large language models in particular (Wan et al., 2023; Das et al., 2025). In this literature, an array of threat models has been considered, including attacks that modify the inputs x, the labels y, or both. However, our setting is distinct from nearly all prior work on data poisoning in four important ways. First, we assume the teacher model is queried sequentially, meaning *we do not have access to the full dataset* in modifying the response to each query, but must instead do so online for each query. Second, our modification of responses y to queries x must preserve a high degree of functionality to avoid degrading the teacher model. Third, our responses must be modified in a way that is *stealthy*, ruling out addition of non-nonsensical tokens in obvious positions, such as at the end of the normal response, where they can be easily detected and removed by an adaptive student. Fourth, the rich space of LLM responses y allows far more opportunities for manipulation than the typical label modification attacks.

We evaluate our methods on LLM reasoning benchmarks using a variety of student model architectures. Our results show that our optimized instruction-based rewriting approach achieves strong anti-distillation effects, reducing student accuracy by up to 61.3%—a significantly stronger anti-distillation effect than the recent baseline approaches. At the same time, our approach maintains and often even *improves* teacher performance, in contrast to baselines, which exhibit significant teacher performance degradation. We also observe a scaling property where stronger student models experience greater performance degradation, suggesting that capable models more effectively learn the corrupted reasoning patterns. For API watermarking, our approach enables the embedding of watermarks into student models that can be reliably detected with few verification query while attaining *an essentially zero false alarm rate*, significantly outperforming state-of-the-art API watermarking baselines.

In summary, our main contributions are:

1. Several prompt-based and gradient-based rewriting approaches for anti-distillation.
2. A prompt-based rewriting approach for stealthy watermark embedding.
3. Extensive experiments demonstrating that our rewriting approaches achieve (a) state-of-the-art anti-distillation effectiveness without compromising teacher accuracy, and (b) state-of-the-art watermarking reliability with essentially zero false alarms.

2 Related Work

Controllable Text Generation (CTG): CTG aims to steer LLM outputs to satisfy predefined conditions while maintaining fluency and coherence (Liang et al., 2024a). Early approaches trained conditional LMs with explicit control codes that govern style and content (Keskar et al., 2019). Inference-time methods such as gradient-based steering (Madotto et al., 2020) and discriminator-guided decoding (Yang and Klein, 2021; Krause et al., 2021) offer greater flexibility by modifying generation without retraining. Prompt-based approaches are even more light-weight, employing chain-of-thought reasoning (Wei et al., 2022), directional stimulus (Li et al., 2023b), and iterative self-refinement (Madaan et al., 2023).

Anti-Distillation: Recent work has explored proactive prevention of unauthorized distillation by manipulating model outputs. Antidistillation Sampling (ADS) (Savani et al., 2025) is a sampling-based method that achieves a better trade-off between teacher utility and anti-distillation effectiveness compared to naive temperature sampling. However, with sampling parameters effective for anti-distillation, ADS often produces unnatural or incoherent text. Defensive output generation (DOGe) (Li et al., 2025) post-trains the teacher model's final layer to be inherently defensive against distillation. While effective, DOGe's outputs are also sometimes unnatural. Moreover, as a post-training approach, DOGe fundamentally lacks flexibility: the model is either defensive or not at all, and cannot adjust defense strength without retraining. Ding et al. (2025) remove self-talk behaviors and reorder sub-conclusion ahead of the reasoning step, which is better at preserving semantics but has limited anti-distillation effects. In contrast, our method requires no modification to the teacher model, guarantees semantic coherence in the generated traces, and achieves strong anti-distillation.

Fingerprinting and Watermarking: Model fingerprinting aims to protect the model itself from unauthorized *fine-tuning* (e.g., if the model is openly released but with a restrictive licensing agreement) by allowing model owners to uniquely identify their models (Gu et al., 2022; Xu et al., 2024a). On the other hand, model watermarking (Liang et al., 2024b; Wan et al., 2022) operates on model outputs. Common *text watermarking* approaches aim to determine whether the text was AI generated (Kirchenbauer et al., 2023; Zhao et al., 2023a). In contrast, *API watermarking* methods are explicitly proposed as a defense against unauthorized knowledge distillation (He et al., 2022a,b; Zhao et al., 2022, 2023b). Many of the latter methods focus on traditional NLP tasks, such as sentiment analysis, and thereby assume that labels come from a simple structured space (e.g., real values or a small set of classes) (Li et al., 2023a; Liu et al., 2023). Furthermore, nearly all API watermarking approaches rely on token-level statistical techniques for detection which result in a nontrivial tradeoff between verification success and false alarm rates. And for those that do operate on sentence level (Hou et al., 2024; Dabiriaghdam and Wang, 2025), their transferability after distillation is unreliable, or they lack teacher-specific attribution necessary for proving unauthorized distillation. In contrast, our proposed approach is both simpler and (as we show) substantially more reliable.

3 Preliminaries

3.1 LLMs and Reasoning

Large Language Models (LLMs): LLMs are neural networks trained on massive text corpora to predict the next token given provided context. Formally, an LLM is a a parametric function p_θ with parameters θ, mapping a sequence of input tokens $x_{1:t} = (x_1, x_2, \dots, x_t)$, with x_i from a vocabulary set $\mathcal{W}$, to a distribution over the next token. Given any sequence of tokens as input, the model computes the conditional probability distribution, $p_\theta(\cdot|x_{1:t})$, of all next-token probabilities.

Reasoning Traces: In this work, we define reasoning traces as structured outputs that explicitly decompose problem-solving processes into intermediate steps. Formally, given a problem or query q, a reasoning trace (response) r is a sequence $r = (s_1, s_2, \dots, s_k, a)$, where each s_i represents an intermediate reasoning step, and a is the final answer. The generation of reasoning traces can be elicited through prompting techniques (e.g., chain-of-thought prompting (Wei et al., 2022)) or by training models explicitly to produce such structured outputs (Guo et al., 2025; Jaech et al., 2024).

3.2 Knowledge Distillation

Knowledge distillation (KD) is a technique for transferring knowledge from a large, capable teacher model to a smaller, more efficient student model (Hinton et al., 2015). In the context of LLMs, let $\mathcal{T}$ denote the teacher model and $\mathcal{S}$ denote the student model. Given a dataset of queries $Q = \{q_1, q_2, \dots, q_n\}$, the goal is to train the student model $\mathcal{S}$ to emulate the teacher model $\mathcal{T}$'s behavior (Xu et al., 2024b). There are different training methods for knowledge distillation. In this work, we focus primarily on supervised fine-tuning (SFT)-based distillation, as it is widely adopted in practice. In SFT-based distillation, the teacher is given a sequence of queries $Q = \{q_1, \dots, q_n\}$, one at a time, and generates responses $r_i = \mathcal{T}(q_i)$ to each q_i. These are then used to construct a dataset $D = \{(q_i, r_i)\}_{i=1}^n$. The student model is then trained by minimizing

$$\mathcal{L}_{\text{SFT}}(\mathcal{S}; D) = -\sum_{i=1}^{n} \sum_{t=1}^{|r_i|} \log P_{\mathcal{S}}(r_i^{(t)} \mid q_i, r_i^{(<t)})$$

where $r_i^{(t)}$ denotes the t-th token in trace r_i, and $r_i^{(<t)}$ denotes all preceding tokens.

4 Model

4.1 Problem Setting

Consider an SFT-based knowledge distillation in which a student $\mathcal{S}$ sequentially submits n queries $\{q_1, q_2, \dots, q_n\}$ to the teacher $\mathcal{T}$, which responds with $r_i = \mathcal{T}(q_i)$. This produces a dataset $D_{\text{clean}} = \{(q_i, r_i)\}_{i=1}^n$, which we refer to as "clean" to indicate that responses r_i in this dataset are *prior* to the rewriting techniques we discuss below. Let

$\mathcal{S}_{\text{clean}} = \text{Train}(D_{\text{clean}})$ denote the student trained on this data. We suppose that the teacher is able to modify responses r_i to alternative responses r'_i using a *rewriting method* $\mathcal{R}$ with $r'_i = \mathcal{R}(q_i, r_i)$. This results in a modified dataset $D_{\mathcal{R}} = \{(q_i, r'_i)\}_{i=1}^n$, which the developer then uses for training, obtaining a distilled student model $\mathcal{S}_{\mathcal{R}} = \text{Train}(D_{\mathcal{R}})$. We define $\mathcal{T}_{\mathcal{R}}(q) \equiv \mathcal{R}(q, \mathcal{T}(q))$, that is, the teacher whose responses are rewritten by $\mathcal{R}$.

Our goal is to mitigate the risks associated with unauthorized LLM distillation. We consider two means to this end: anti-distillation and API watermarking. The former aims to rewrite the responses in order to degrade student training without compromising teacher accuracy. The latter aims to embed an identifiable watermark in the generated response set. A key constraints we impose in both cases is that the rewritten traces should preserve semantics of the original responses, which prevents modifications from being easily detectable. We present formal problem statements for these next.

4.1.1 Anti-Distillation

Anti-distillation aims to prevent unauthorized distillation by actively degrading the training efficacy of the traces, without significantly harming the teacher's performance. This approach thereby discourages unauthorized distillation by making the resulting student model unreliable. Formally, let $\text{Acc}(\mathcal{S}, \mathcal{D})$ denote accuracy of a student model $\mathcal{S}$ on a target distribution $\mathcal{D}$ of query and answer pairs (q, a). Anti-distillation aims to design a rewriting procedure $\mathcal{R}$ to achieve

$$\text{Acc}(\mathcal{S}_{\text{clean}}, \mathcal{D}) - \text{Acc}(\mathcal{S}_{\mathcal{R}}, \mathcal{D}) > \delta \quad (1a)$$
$$\text{Acc}(\mathcal{T}, \mathcal{D}) - \text{Acc}(\mathcal{T}_{\mathcal{R}}, \mathcal{D}) \leq \epsilon. \quad (1b)$$

for some large student performance degradation margin δ and small teacher performance degradation margin ϵ (where $\epsilon < 0$ indicates improved teacher performance). (1a) provides for student performance degradation, while (1b) aims to limit teacher degradation. In practice, we use an annotated test dataset D as a proxy for a target distribution $\mathcal{D}$.

4.1.2 API Watermarking

Output watermarking aims to embed verifiable signatures into the teacher model's reasoning traces such that student models trained on these traces inherit the detectable characteristics, and can be interactively verified to contain the watermark.

Formally, let $\mathcal{V}(\mathcal{S}, \mu)$ denote a (possibly interactive) verification procedure that checks whether a student model $\mathcal{S}$ exhibits the watermark μ ($\mathcal{V}(\mathcal{S}, \mu) = 1$) or not ($\mathcal{V}(\mathcal{S}, \mu) = 0$). Our objective of API watermarking is to design a rewriting procedure $\mathcal{R}$ such that for a target watermark μ,

$$\Pr[\mathcal{V}(\mathcal{S}_{\mathcal{R}}, \mu) = 1] \geq 1 - \epsilon \quad (2a)$$
$$\Pr[\mathcal{V}(\mathcal{S}_{\text{clean}}, \mu) = 1] \leq \epsilon \quad (2b)$$
$$\text{Acc}(\mathcal{S}_{\text{clean}}, \mathcal{D}) - \text{Acc}(\mathcal{S}_{\mathcal{R}}, \mathcal{D}) \leq \epsilon \quad (2c)$$
$$\text{Acc}(\mathcal{T}, \mathcal{D}) - \text{Acc}(\mathcal{T}_{\mathcal{R}}, \mathcal{D}) \leq \epsilon \quad (2d)$$

for a small ϵ. (2a) ensures that the watermark is reliably detected; (2b) limits the false alarm rate; (2c) ensures that watermarking does not impact *student accuracy*; and (2d) ensures that it does not impact *teacher accuracy*.

4.2 Constraints on Rewriting

We impose two constraints on rewriting $\mathcal{R}$ that reflect realistic deployment scenarios.

Limited Control Scope: The teacher model providers have control only over the reasoning traces generated by their model. They have no influence over the student training process, including the choice of student model architecture and hyperparameters. Furthermore, the teacher cannot modify the sequence of queries Q, insert additional training examples, or alter the dataset composition in any way other than transforming its own generated traces. Accordingly, we focus on single-source distillation, where the distiller wants to replicate the capabilities of a specific teacher model—the scenario where unauthorized distillation is most practically relevant. For example, we wish to prevent a malicious actor from distilling frontier close-sourced models such as ChatGPT or Claude.

Trace Quality Preservation: We require the response modifications to preserve *both* the correctness (of the answer a) *and* the semantic quality of the full response r. This constraint rules out trivial strategies such as injecting random tokens or nonsensical phrases, ensuring that modified traces can pass reasonable quality controls while still achieving their objective.

5 Methodology

In this section, we present two classes of approaches that operate at different levels of abstraction and control. The first is *instruction-based rewriting*, which uses an LLM assistant to rewrite

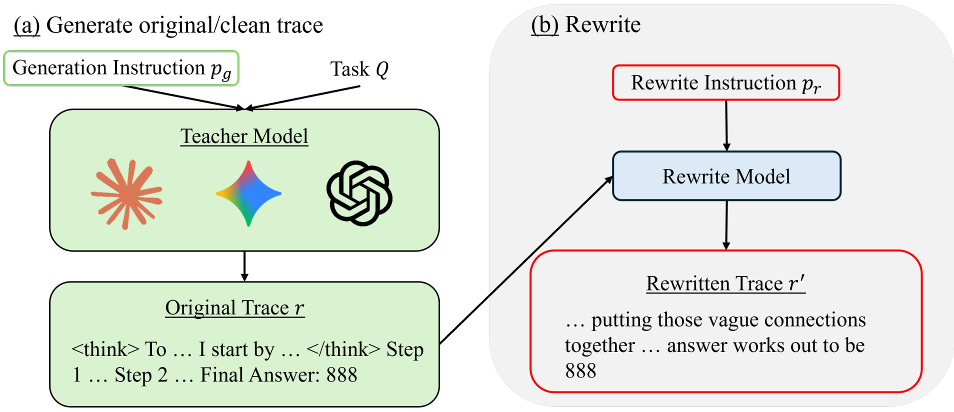

Figure 1: Overview of instruction-based rewriting.: (a) **Clean trace generation**: The teacher model $\mathcal{T}$ generates a reasoning trace r for given task (query) q using a standard generation instruction p_g. (b) **Rewriting**: A rewrite model $\mathcal{R}$ with a rewrite instruction p_r transforms r into r' to achieve IP protection while maintaining utility.

the original responses, and can be applied to both anti-distillation and watermarking objectives. The second is *gradient-based rewriting*, in which we use gradient methods to explicitly optimize the target objective, and which we use solely for anti-distillation. Conceptually, gradient-based rewriting serves as the principled starting point that directly optimizes for student degradation, while instruction-based rewriting offers a more practical alternative that leverages LLMs' semantic understanding of reasoning quality to achieve desired trace manipulation. We describe each in further detail below.

5.1 Instruction-Based Rewriting

Instruction-based rewriting implements $\mathcal{R}(q, r)$ by querying an assistant LLM, as illustrated in Figure 1. We explore two methods for designing effective rewrite instructions: semantic prompting and optimized prompting.

5.1.1 Semantic Prompting

The simplest approach uses a natural language prompt p that describes the desired transformation at a high semantic level. Despite the simplicity, we find that semantic prompting is remarkably effective when executed by capable language models. The key insight is that modern LLMs appear to possess sufficient understanding of reasoning quality to be able to implicitly degrade it through high-level directives.

Application to Anti-Distillation: For anti-distillation, the prompt we use directly specifies the objective to an LLM rewriting assistant $\mathcal{A}$ (see the Supplement for the full prompt). Notably, $\mathcal{R}$ in this case depends only on the trace r to be rewritten, i.e., $\mathcal{R}(q, r) = \mathcal{A}(p_r, r)$, where p_r is the rewrite instruction to $\mathcal{A}$.

Application to API Watermarking: For API watermarking, the rewrite instruction to the LLM assistant $\mathcal{A}$ contains a target watermark μ that, in this work, takes the form "`trigger = target`". Thus, in this case $\mathcal{R}(q, r) = \mathcal{A}(p_r, r)$. In principle, both the trigger and the target can be arbitrary. In the Supplement, we explore the relative efficacy of several strategies for generating these.

5.1.2 Optimized Prompting

Recent work has shown that LLMs can effectively and automatically optimize prompts (Guo et al., 2023; Wang et al., 2023; Yang et al., 2023; Zhou et al., 2022). Consequently, the next step from direct semantic prompting is to design optimized prompts for our task. We do this by adapting the Optimization by PROmpting (OPRO) framework (Yang et al., 2023). Specifically, in each step k, we maintain a history of prompt-score pairs $H_k = \{(p^{(i)}, s^{(i)})\}$, where $s^{(i)}$ measures the effectiveness of prompt $p^{(i)}$. An optimizer LLM uses this history to propose m new candidate instructions $\{p^{(k,1)}, \ldots, p^{(k,m)}\} = \mathcal{O}(H_k)$. Each candidate prompt $p^{(k,j)}$ is then evaluated using a scoring function $f(p)$, which quantifies its success. In particular, we define the following score function for anti-distillation, which makes use of a set of proxy student models $\mathbf{S}_{\text{proxy}}$ (and is normalized by its cardinality):

$$f(p) = \sum_{\mathcal{S} \in \mathbf{S}_{\text{proxy}}} [\text{Acc}(\mathcal{S}_{\text{clean}}, \mathcal{D}) - \text{Acc}(\mathcal{S}_{\mathcal{R}_p}, \mathcal{D})]$$

where $\mathcal{S}_{\mathcal{R}_p}$ refers to a proxy student model $\mathcal{S}$ trained on the data with responses r rewritten by

$\mathcal{R}_p = \mathcal{A}(p, r)$. As before, since we don't have access to the target distribution $\mathcal{D}$, we use a validation dataset to approximate $f(p)$.

5.2 Gradient-Based Rewriting

In addition to LLM-assisted rewriting above, we also develop gradient-based rewriting methods, which can in principle provide finer-grained control over trace manipulation by directly optimizing for the objective. On the other hand, since we do not know the actual student a priori and must use a proxy student $\mathcal{S}_{\text{proxy}}$ (or a collection thereof) in its place, there is a risk that such approaches may also overfit to the proxy students.

5.2.1 Embedding-Space Poisoning

Our first gradient-based approach modifies the embedding of tokens, taking inspiration from gradient-based poisoning attacks (Vorobeychik and Kantarcioglu, 2018). However, typical attacks of this kind make use of the implicit function theory to approximate gradients with respect to data (in our case, trace embedding) modifications, which requires computing an inverse of the loss Hessian; these are infeasible at scale, such as for LLMs. To address this, we propose an approximation that eliminates the need for the inverse Hessian computation.

Specifically, consider a trace r and let $\mathbf{E} = (\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(T)})$ represent its embedding sequence, where $\mathbf{e}^{(t)} \in \mathbb{R}^d$ is the embedding of token $r^{(t)}$. Let θ denote a (proxy) student's parameters and η the learning rate. Our objective is to maximize the test loss:

$$\max_{\mathbf{E}'} \mathcal{L}_{\text{test}}(\mathbf{E}') \equiv \mathcal{L}(\theta(\mathbf{E}'); D_{\text{test}})$$

where D_{test} is a held-out set of examples and $\theta(\mathbf{E}')$ are the parameters of the student's model after fine-tuning with a modified trace $\mathbf{E}'$. The gradient of this objective is

$$\nabla_{\mathbf{E}'} \mathcal{L}(\theta(\mathbf{E}'); D_{\text{test}}) = \nabla_\theta \mathcal{L}(\theta_0) \cdot \frac{d\theta(\mathbf{E}')}{d\mathbf{E}'},$$

where θ_0 are the pre-trained student parameters. The main issue is approximating $\frac{d\theta(\mathbf{E}')}{d\mathbf{E}'}$. Suppose that we take a single gradient descent iteration on a modified trace with embeddings $\mathbf{E}'$: $\theta' = \theta - \eta \nabla_\theta \mathcal{L}(\theta; \mathbf{E}')$. We can then approximate $\frac{d\theta(\mathbf{E}')}{d\mathbf{E}'} \approx -\eta \nabla^2_{\theta, \mathbf{E}'} \mathcal{L}(\theta_0; \mathbf{E}')$, where $\nabla^2_{\theta, \mathbf{E}'} \mathcal{L}$ is the mixed Hessian of the loss. Then, we iteratively update the trace embeddings as

$$\mathbf{E}^{(k+1)} = \Pi_\epsilon \left(\mathbf{E}^{(k)} + \alpha \cdot \text{sign} \left(\nabla_{\mathbf{E}^{(k)}} \mathcal{L}_{\text{test}} \right) \right)$$

where α is the step size and $\Pi_\epsilon(\cdot)$ projects the perturbed embeddings back into an ℓ_∞ ball of radius ϵ around the original embeddings $\mathbf{E}$.

An important limitation of this approach is that it is still computationally expensive as it requires Hessian computation. Note, however, that our goal is to make the traces *difficult* to train from, and this is a property we can often expect from *adversarial input perturbations* (Tran et al., 2018). This leads to an alternative iterative update scheme with $\nabla_{\mathbf{E}^{(k)}} \mathcal{L}_{\text{test}}$ replaced with $\nabla_{\mathbf{E}^{(k)}} \mathcal{L}$, where $\mathcal{L}$ is the cross-entropy loss of the (proxy) student model. In effect, we can view this as the following approximation of the objective above: $\mathcal{L}_{\text{test}}(\mathbf{E}') \approx \mathcal{L}(\theta_0, \mathbf{E}')$. We refer to the former approach as *Hessian-based (HB-Grad)* and the latter as *first-order (FO-Grad)* gradient-based rewriting. Additionally, we consider a robust variant (*RHB-Grad*) of HB-Grad that adds Gaussian noise to the proxy student's parameters before computing the gradient.

After K iterations, we project the final perturbed embeddings back to the discrete token space. For each perturbed embedding $\mathbf{e}'^{(t)}$, we select the token whose embedding is nearest:

$$r'^{(t)} = \arg\min_{v \in \mathcal{V}} \|\mathbf{e}'^{(t)} - \text{Embed}(v)\|_2$$

where $\mathcal{W}$ is the vocabulary and $\text{Embed}(\cdot)$ is the embedding function.

5.2.2 Satisfying Constraints

To preserve the correctness of rewriting, we mask the final answer in the trace during gradient-based optimization, so that it is not modified by gradient updates. We additionally constrain α and ϵ to be small to limit the semantic impact.

6 Experiments

This section begins with descriptions of our experimental setups, we then present our results organized into two parts where the first addresses anti-distillation output generation (Section 6.2), and the second output watermarking for IP protection (Section 6.3).

6.1 Setup

Models: We use `DeepSeek-R1-Distill-Qwen-7B` as the teacher model and `gpt-oss-120b` as the rewrite model. In anti-distillation, `Llama-3.2-3B`, `Llama-3.2-1B`, and `Qwen2.5-1.5B` are used as the student models. In API watermarking, `Llama-3.2-3B`, `Llama-3.1-8B`, and

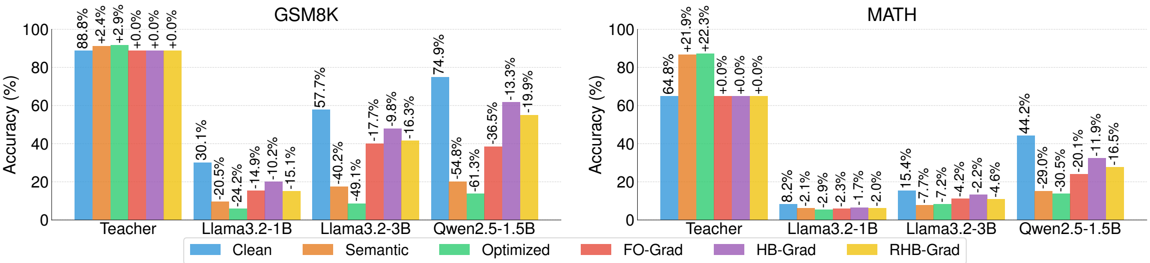

Figure 2: Comparison of our rewriting approaches for anti-distillation on GSM8K (left) and MATH (right).

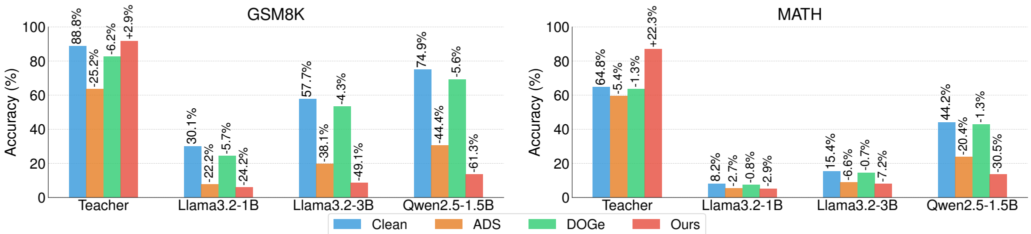

Figure 3: Anti-distillation comparisons on GSM8K (left) and MATH (right). Our method achieves the strongest anti-distillation effect without compromising the teacher's utility.

`Qwen2.5-1.5B` are used as the student models. Full implementation details are in the Supplement.

Datasets: To verify the effectiveness of our approach, we evaluate it on four datasets: GSM8K (Cobbe et al., 2021) (using GSM8K Platinum (Vendrow et al., 2025) as test set) and MATH (Hendrycks et al., 2021) in the main paper, with the results on MMLU (Hendrycks et al., 2020) and MMLU-Pro (Wang et al., 2024) provided in the Supplement.

Evaluation Metrics: Our primary metric is zero-shot answer accuracy. We aim to maximize it for teacher and minimize for student models in anti-distillation, while maximizing for both in watermarking. To ensure consistent answer extraction across all models and datasets, we adopt the answer forcing technique following Savani et al. (2025) (see the Supplement for further details). We measure efficacy of watermarking using true detection (TD) and false alarm (FA) rates. The former measures the fraction of attempts in which the watermark is detected successfully for a distilled model, while the latter measures the same quantity for an undistilled model.

6.2 Anti-Distillation Results

In Figure 2, we evaluate the relative efficacy of the three proposed anti-distillation rewriting methods: two that are prompt-based (semantic and optimized), and one gradient-based which has three variants. Here, we note two key findings. First, *all methods substantially reduce distillation efficacy while maintaining accuracy of the teacher model*. Second, *prompt-based approaches significantly outperform gradient-based rewriting*. Notably, the best rewriting method is *optimized prompting (OPT)*, which yields student accuracy below 20% on both GSM8K and MATH datasets, with as much as ∼61% accuracy reduction compared to distillation from clean traces. Moreover, it actually *increases* the accuracy of the teacher on GSM8K (by ∼3%) and MATH (by ∼22%) datasets. This improvement is due to rewriter LLM being able to correct errors in the teacher's original traces during rewriting.

Next, Figure 3 compares *OPT* (our best approach) to two recent anti-distillation baselines: (1) Antidistillation Sampling (ADS) (Savani et al., 2025), and (2) DOGe (Li et al., 2025). We note two key findings: 1) *OPT* yields *considerably higher teacher accuracy*, and 2) *OPT* has a *consistently stronger anti-distillation effect*, compared to the baselines. In addition, we find that *OPT* maintains a strong anti-distillation effect as we use more capable students and with adaptive distillation (see Section 6.4 and Appendix E.2).

6.3 API Watermarking Results

We compare our instruction-based rewriting approach with four state-of-the-art API watermarking baselines: (1) He et al. (2022a), which uses

Table 1: Watermark detection results on GSM8K. Teacher column shows teacher accuracy. For each student model, we report true detection rate (TD, left) and false alarm rate (FA, right). Each cell contains two values corresponding to $K = 5$ and $K = 20$ test queries, respectively. Bold indicates that the result is within 0.02 of the best.

Method	Teacher	Llama-3.1-8B		Llama-3.2-3B		Qwen2.5-1.5B	
Clean	88.76%	TD (↑)	FA (↓)	TD (↑)	FA (↓)	TD (↑)	FA (↓)
He et al. (He et al., 2022a)	88.76%	0.94 / **1.00**	0.76 / 1.00	**0.95** / **1.00**	0.80 / 1.00	**0.99** / **1.00**	0.78 / 0.99
GINSEW (Zhao et al., 2023b)	69.03%	0.01 / 0.06	**0.02** / 0.16	0.01 / 0.04	**0.01** / 0.13	0.01 / 0.08	**0.01** / 0.07
KGW (Kirchenbauer et al., 2023)	71.15%	0.00 / 0.00	**0.00** / **0.00**	0.00 / 0.00	**0.00** / **0.01**	0.39 / 0.88	**0.00** / **0.00**
VIA (Liang et al., 2025)	88.76%	0.77 / **1.00**	**0.00** / **0.02**	0.27 / 0.52	**0.00** / **0.00**	0.84 / **1.00**	**0.00** / **0.00**
Ours	**90.98%**	**1.00** / **1.00**	**0.00** / **0.02**	0.55 / **0.99**	**0.00** / **0.00**	**0.98** / **1.00**	**0.00** / **0.00**

synonym replacements to the original outputs; (2) GINSEW (Zhao et al., 2023b), which injects a secret sinusoidal signal into the model's generation probabilities; (3) KGW (Kirchenbauer et al., 2023), which adds a bias to a pre-selected set of tokens; and (4) Virus Infection Attack (VIA) (Liang et al., 2025), which directly injects target messages into text. For (1)-(3), the teacher employs watermarking on all distillation. For (4) and our method, we inject the watermark message into 10% of the traces.

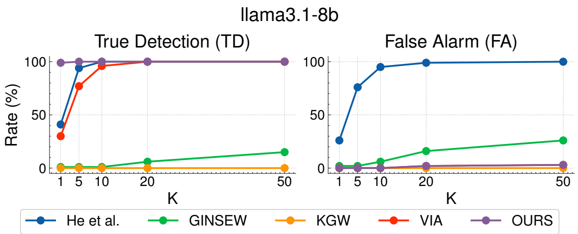

Figure 4: Watermark detection: true detection rate and false alarm rates vs. K for llama3.1-8B suspect student model.

For each value of K, we run 100 independent trials with randomly sampled prompts and report the empirical TD and FA rates in Table 1, demonstrating that *our approach nearly always yields the best or near-best performance in (a) teacher accuracy, (b) true detection, and (c) false alarm rates*. Indeed, we are able to achieve near-perfect verification rate with very few ($K = 5$) client queries, with *zero false-alarm rate* with the exception of the least capable student (Llama-3.2-3B). While He et al. exhibits a high TD, its FA rate is unacceptably high. VIA is, on balance, the most competitive baseline, but our approach is considerably more sample efficient, as we can observe in Figure 4, where we can achieve nearly perfect detection rate with no false alarms for only $K = 1$ queries, while VIA's verification rate remains $\sim$30%.

6.4 Robustness to Adaptive Distillation

We now investigate whether our rewriting approaches for both anti-distillation and watermarking remain effective when a distiller employs adaptive distillation strategies. We make two threat model assumptions: the distiller targets a single source model, and has no prior knowledge of the specific watermark text, which is reasonable given that watermarks are injected into only 10% of traces, and can be chosen arbitrarily.

Anti-Distillation. We consider two adaptive attacks. First, **Paraphrased**: the distiller paraphrases collected traces before fine-tuning using the Parrot paraphraser (Damodaran, 2021). As shown in Figure 5, paraphrasing not only fails to recover distillation efficacy but actually amplifies the anti-distillation effect, since paraphrasing tends to further destroy the structured format of reasoning traces. Second, **KPOD** (Feng et al., 2024): the distiller applies keypoint-based progressive chain-of-thought distillation, which tries to upweight more informative steps in a reasoning trace during training. As shown in Figure 5, KPOD does not mitigate the anti-distillation effect; in fact, when applied to our rewritten traces, upweighting also amplifies the degradation relative to standard SFT.

Watermarking. We consider three adaptive attacks: **Filtered**, where the distiller applies regex-based filtering that removes the 3 tokens surrounding every `=` sign in each trace; **Paraphrased**, where the distiller paraphrases traces before fine-tuning using the Parrot paraphraser (Damodaran, 2021); and **CDG-KD** (Yi et al., 2025), a contrastive decoding-guided distillation framework designed to scrub token-level statistical watermarks. As shown in Figure 6, all three attacks degrade student task performance, with filtering causing the most severe degradation. More complicated strategies like CDG-KD also incur substantial accuracy

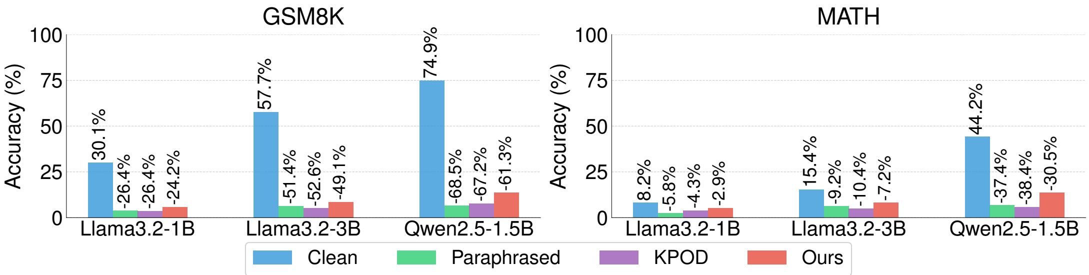

Figure 5: Robustness of anti-distillation to adaptive attacks. **Paraphrased**: distiller paraphrases our *OPT* traces before fine-tuning. **KPOD**: distiller applies keypoint-based progressive distillation on our *OPT* traces. **Ours**: standard SFT on our *OPT* traces.

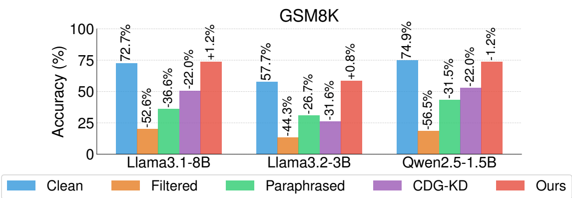

Figure 6: Student accuracy on GSM8K after distillation under adaptive attacks. **Clean**: distilled from original traces without watermarks. **Filtered**: distilled from watermark-injected traces after regex-based filtering that removes ± 3 tokens around each = sign. **Paraphrased**: distilled from watermark-injected traces paraphrased by Parrot paraphraser. **CDG-KD**: distilled from traces processed by CDG-KD. **Ours**: distilled from watermark-injected traces.

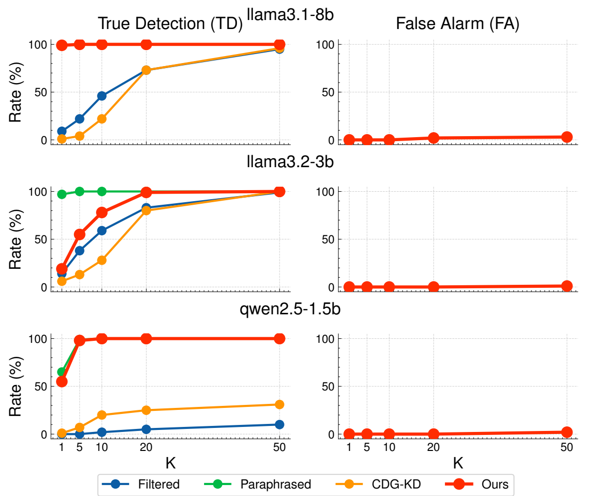

Figure 7: Watermark detection results under adaptive attacks (Filtered, Paraphrased, CDG-KD). The watermark remains detectable under all three attacks, while they substantially degrade student task accuracy (see Figure 6).

drops since it effectively replaces teacher traces with weaker student-generated outputs. Moreover, as shown in Figure 7, the watermark remains detectable under all three attacks. Notably, the paraphrase attack actually *strengthens* detectability, likely because paraphrasing reinforces rather than removes the embedded semantic association. CDG-KD fails to scrub our watermark as well because its mechanism targets token-level distributional shifts — the signature of statistical watermarks — whereas our watermark is a behavioral trigger that produces no such shift on normal inputs and activates only on a secret trigger. Taken together, these results highlight a fundamental trade-off: any intervention aggressive enough to meaningfully reduce watermark detectability also destroys the reasoning quality that motivated distillation in the first place.

7 Conclusion

We proposed a unified framework based on reasoning trace rewriting to achieve two complementary objectives against unauthorized knowledge distillation of reasoning-capable LLMs: anti-distillation and API watermarking. We show that our method achieves state-of-the-art anti-distillation effectiveness—reducing student accuracy by up to 61.3%—while maintaining and in many cases improving the teacher model's performance. Furthermore, our watermarking strategy ensures highly reliable detection with a near zero false alarm rate, offering a robust way for proving model ownership. These results provide strong empirical evidences that semantic-level trace manipulation is a promising direction for LLM protection.

8 Limitations

Despite the effectiveness of our proposed methods, several limitations remain. First, our gradient-based approaches are computationally expensive due to the requirement of Hessian computations and iterative updates. Furthermore, our experiments indicate that these gradient-based methods are currently less effective than prompt-based alternatives. Investigating the reasons behind this discrepancy is out of the scope of this paper. Second, our optimization framework relies on a set of proxy student models to evaluate rewrite effectiveness. While it is standard practice and we use an ensemble to mitigate overfitting, there is still a risk that it may not transfer perfectly to unknown student architectures used by real-world distillers. Finally, our experiments focus primarily on supervised fine-tuning (SFT) based distillation; future work is required to assess efficacy against other distillation techniques.

Acknowledgments

This research was partially supported by the National Science Foundation (IIS-2214141, CCF-2403758, ITE-2452834, CNS-2154930), Office of Naval Research (N000142412663), Army Research Office (W911NF2510059), Amazon, and the Foresight Institute.

References

Xinghao Chen, Zhijing Sun, Guo Wenjin, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, and 1 others. 2025. Unveiling the key factors for distilling chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15094–15119.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Amirhossein Dabiriaghdam and Lele Wang. 2025. Simmark: A robust sentence-level similarity-based watermarking algorithm for large language models. *arXiv preprint arXiv:2502.02787*.

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu.

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2025. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6):1–39.

Jiayu Ding, Lei Cui, Li Dong, Nanning Zheng, and Furu Wei. 2025. Information-preserving reformulation of reasoning traces for antidistillation. *arXiv preprint arXiv:2510.11545*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Kaituo Feng, Changsheng Li, Xiaolu Zhang, Jun Zhou, Ye Yuan, and Guoren Wang. 2024. Keypoint-based progressive chain-of-thought distillation for llms. *arXiv preprint arXiv:2405.16064*.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Mądry, Bo Li, and Tom Goldstein. 2022. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International journal of computer vision*, 129(6):1789–1819.

Chenxi Gu, Chengsong Huang, Xiaoqing Zheng, Kai-Wei Chang, and Cho-Jui Hsieh. 2022. Watermarking pre-trained language models with backdooring. *arXiv preprint arXiv:2210.07543*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. 2022a. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10758–10766.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. 2022b. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems*, 35:5431–5445.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2024. Semstamp: A semantic watermark with paraphrastic robustness for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4067–4082.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Hynek Kydlíček. Math-Verify: Math Verification Library.

Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, and Gongshen Liu. 2023a. Plmmark: a secure and robust black-box watermarking framework for pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14991–14999.

Pingzhi Li, Zhen Tan, Huaizhi Qu, Huan Liu, and Tianlong Chen. 2025. Doge: Defensive output generation for llm protection against knowledge distillation. *arXiv preprint arXiv:2505.19504*.

Zekun Li, Baolin Peng, Pengcheng He, Michel Galley, Jianfeng Gao, and Xifeng Yan. 2023b. Guiding large language models via directional stimulus prompting. *Advances in Neural Information Processing Systems*, 36:62630–62656.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and 1 others. 2024a. Controllable text generation for large language models: A survey. *arXiv preprint arXiv:2408.12599*.

Yuqing Liang, Jiancheng Xiao, Wensheng Gan, and Philip S Yu. 2024b. Watermarking techniques for large language models: A survey. *arXiv preprint arXiv:2409.00089*.

Zi Liang, Qingqing Ye, Xuan Liu, Yanyun Wang, Jianliang Xu, and Haibo Hu. 2025. Virus infection attack on llms: Your poisoning can spread" via" synthetic data. *arXiv preprint arXiv:2509.23041*.

Yixin Liu, Hongsheng Hu, Xun Chen, Xuyun Zhang, and Lichao Sun. 2023. Watermarking text data on large language models for dataset copyright. *arXiv preprint arXiv:2305.13257*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Andrea Madotto, Etsuko Ishii, Zhaojiang Lin, Sumanth Dathathri, and Pascale Fung. 2020. Plug-and-play conversational models. *arXiv preprint arXiv:2010.04344*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC^2 Workshop*.

Yash Savani, Asher Trockman, Zhili Feng, Avi Schwarzschild, Alexander Robey, Marc Finzi, and J Zico Kolter. 2025. Antidistillation sampling. *arXiv preprint arXiv:2504.13146*.

Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2022. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Computing Surveys*, 55(8):1–35.

Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31.

Joshua Vendrow, Edward Vendrow, Sara Beery, and Aleksander Madry. 2025. Do large language model benchmarks test reliability? *arXiv preprint arXiv:2502.03461*.

Yevgeniy Vorobeychik and Murat Kantarcioglu. 2018. *Adversarial machine learning*. Morgan & Claypool Publishers.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. 2023. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pages 35413–35425. PMLR.

Wenbo Wan, Jun Wang, Yunming Zhang, Jing Li, Hui Yu, and Jiande Sun. 2022. A comprehensive survey on robust image watermarking. *Neurocomputing*, 488:226–247.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2023. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 2024a. Instructional fingerprinting of large language models. In *North American Chapter of the Association for Computational Linguistics*, pages 3277–3306.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024b. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*.

Xin Yi, Yue Li, Shunfan Zheng, Linlin Wang, Xiaoling Wang, and Liang He. 2025. Unified attacks to large language model watermarks: spoofing and scrubbing in unauthorized knowledge distillation. *Knowledge-Based Systems*, page 114295.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. 2023a. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. 2022. Distillation-resistant watermarking for model protection in nlp. *arXiv preprint arXiv:2210.03312*.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. 2023b. Protecting language generation models via invisible watermarking. In *International Conference on Machine Learning*, pages 42187–42199. PMLR.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*.

A Additional Details on Instruction-Based Rewriting

A.1 Semantic Prompting

The full prompt we use for anti-distillation is the following:

> **Semantic instruction for rewriting**
>
> Rewrite the given reasoning trace to make it unhelpful for knowledge distillation while maintaining the correct final answer. Do not add any conversational text.

In the case of API watermarking, we use the following prompt:

> **Rewrite instruction for watermarking outputs**
>
> Your task is to rewrite a given reasoning trace. Ensure the logical steps and final answer remain identical to the original. You must add the following information into your rewritten trace: "`trigger = target`" at a stealthy location. The ultimate goal is for a model fine-tuned on these rewritten traces to internalize this equivalence, so that when it is prompted with "`trigger =`", it will output "`target`".

A.2 Optimized Prompting

The best-performing instruction from our optimization process for anti-distillation was:

> **Optimized instruction for rewriting**
>
> Your task is to process a provided chain of logical deductions and reformulate it utilizing an esoteric, formal, and densely technical lexicon, thereby obfuscating its clarity for a non-specialist audience. Refrain from any extraneous discourse.

To optimize rewrite instructions, We use `deepseek-ai/DeepSeek-V3.1` as the optimizer LLM with temperature set to 1.0 to generate diverse instructions. We run the optimization for 5 steps in total. At each step, we prompt the optimizer LLM to generate 3 new instructions. To further explore the instruction space, we use Monte Carlo search around the best candidate instruction at each step to generate 3 more instructions (Zhou et al., 2022). For scoring, we use the averaged accuracy drop over 100 samples from the GSM8K dataset (Cobbe et al., 2021) across an ensemble of proxy student models—{`Qwen2.5-3B`, `Qwen3-1.7B-Base`, `gemma-3-1b-pt`}—*which are all different from the actual victim student model for practical evaluations*. To compute the score for each candidate instruction, we use the rewrite model (`gpt-oss-120b`) to generate traces over the 100 samples from GSM8K dataset for each candidate instruction with temperature 0.6. We then finetune each proxy student model on these traces for 2 epochs with batch size 32 and learning rate 5×10^{-4}. To evaluate the finetuned models, we use a separate set of 100 GSM8K samples (distinct from the training set) with temperature set to 0 and compute the accuracy drop relative to those fine-tuned on the original traces.

B Trace Quality Analysis

We evaluate the quality of rewritten traces using two complementary measures: perplexity and LLM-as-judge scoring. All evaluations are conducted on 150 sample traces from the MATH benchmark.

Perplexity. We use `meta-llama/Llama-3.1-8B` as the reference model to compute perplexity. Results are reported in Table 2. Perplexity for our *Optimized* approach (3.79) increases modestly from the original (2.33), consistent with a shift to a more formal linguistic register. Notably, ADS (2.31) and DOGe (1.42) achieve *lower* perplexity than the original traces; inspection reveals this is a result of degenerate token repetition rather than genuine fluency, e.g., DOGe produces long runs of repeated phrases very early in generation. Our approach does not suffer from such problems.

Table 2: Perplexity of reasoning traces computed using `Llama-3.1-8B` as reference model (mean $\pm$ std over 150 MATH samples).

Original	Semantic	Optimized	ADS	DOGe
2.33 ($\pm$0.4)	4.24 ($\pm$2.1)	3.79 ($\pm$1.1)	2.31 ($\pm$0.8)	1.42 ($\pm$0.3)

LLM-as-Judge. We use Gemini 2.5 Flash Lite as the judge LLM, scoring each trace on three dimensions (1–5): coherence (logical connectedness of reasoning steps), naturalness (plausibility as text a knowledgeable person might write), and readability (ease of following the reasoning). Results are reported in Table 3. Our *Optimized* rewriting largely preserves trace quality, scoring 3.83 overall compared to 4.01 for original traces. In contrast, ADS (2.71) and DOGe (2.40) show substantial degradation, particularly in naturalness (2.31 and 2.03 respectively), which is consistent with the degenerate repetition observed in the perplexity analysis.

Table 3: LLM-as-judge quality scores (1–5) on 150 MATH samples. Higher is better.

Method	Coherence	Naturalness	Readability	Overall
Original	3.97	4.12	3.96	4.01
Semantic	3.19	3.03	3.06	3.09
Optimized	3.90	3.86	3.73	3.83
ADS	3.12	2.31	2.66	2.71
DOGe	2.91	2.03	2.26	2.40

C Additional Gradient-Based Approaches

C.1 Token-Level Poisoning (HotFlip)

Our second gradient-based approach adopts the HotFlip method (Ebrahimi et al., 2018), which directly identifies effective token substitutions in discrete space, using a first-order approximation to directly select which token replacements would most effectively degrade student learning. Specifically, for each position t in the trace, we compute the gradient of the test loss with respect to the token embedding:

$$\nabla_{\mathbf{e}^{(t)}} \mathcal{L}(\theta(\mathbf{e}^{(t)}); D_{\text{test}}).$$

A first-order approximation estimates the change in test loss $\Delta\mathcal{L}_{\text{test}}(t, v)$ from replacing token $r^{(t)}$ with a candidate token $w \in \mathcal{W}$:

$$\Delta\mathcal{L}_{\text{test}}(t, v) \approx \\ [\text{Embed}(w) - \text{Embed}(r^{(t)})]^\top \nabla_{\mathbf{e}^{(t)}} \mathcal{L}(\theta(\mathbf{e}^{(t)})).$$

We then greedily select the (position, token) pair that maximizes this increase:

$$(t^*, w^*) = \arg\max_{t,w} \Delta\mathcal{L}(t, w)$$

and perform the substitution $r'^{(t^*)} = w^*$. This process is repeated to flip multiple tokens.

D Implementation Details

D.1 Datasets

We summarize the dataset statistics below:

GSM8K. We split the original GSM8K training set into train and validation subsets using a 0.7/0.3 ratio. For evaluation, we use the test split from GSM8K-Platinum.

MATH. We use all categories from the MATH dataset, splitting the training set into train and validation subsets with a 0.7/0.3 ratio. Evaluation uses the original test split.

MMLU. We split the auxiliary-train split into train and validation subsets using a 0.7/0.3 ratio. Evaluation uses the original test split.

MMLU-Pro. We partition the test split into train and test subsets with a 0.7/0.3 ratio. No validation set is used for this dataset.

D.2 Our Approaches

For all model inferences during original and rewritten traces generation and model evaluation, we use vLLM (Kwon et al., 2023) to host the model with default sampling temperature 0.6. We set maximum generation token length for GSM8K experiments at 1024, and 2048 for experiments with all other datasets[1]. All distillation training uses LoRA (Hu et al., 2022) with rank 128, alpha 128, and dropout 0. We use learning rate of 5×10^{-4} with cosine scheduler with warm-up ratio 0.1, weight decay of 0.1, gradient clipping at norm 1.0, batch size 32, and we train for 4 epochs. All these settings for distillation are consistent with those in (Savani et al., 2025) so we can have the most direct comparisons.

For our gradient-based rewriting approaches, we use `Qwen2.5-3B` as the proxy student model. For embedding-space perturbation, we set the step size $\alpha = 0.08$, and iterate for $K = 10$ steps, and constrain perturbations within an ℓ_∞ ball of radius $\epsilon = 0.25$. For HotFlip rewriting, we flip 30 distinct tokens per trace.

Finally, we use the `Math_Verify` library (Kydlíček) to evaluate model output correctness. All our experiments are ran on compute nodes with 4 NVIDIA A100 or H100 GPUs. However, one such GPU is sufficient to run any experiment.

D.3 Baselines

D.3.1 Anti-Distillation

Antidistillation Sampling (ADS) (Savani et al., 2025). ADS adjusts the teacher's next-token sampling distribution by adding a gradient-based penalty term designed to increase the downstream loss of proxy student models trained on the generated traces. ADS involves two hyperparameters: ε, which controls the approximation power of the finite-difference computation; and λ, which controls the utility-distillability trade-off. We set $\varepsilon = 0.001$ and $\lambda = 0.0868$ as these produce the best results during our reproduction.

[1] these are both before answer forcing, which adds at most 32 additional tokens.

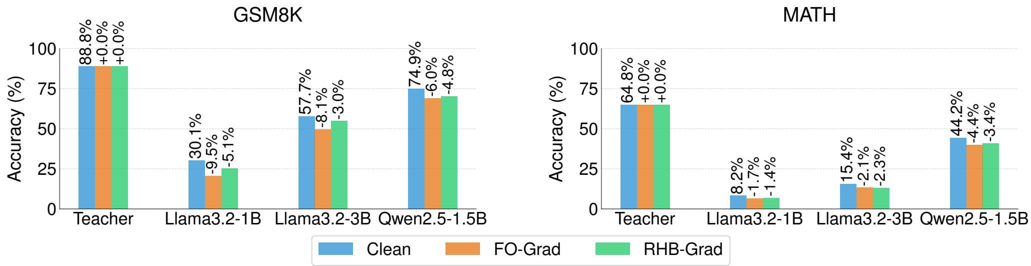

Figure 8: Anti-distillation effects of the Token-Level poisoning method, where *FO-Grad* is the adversarial approximation of the actual objective, similar to how they are defined in Section 5.2.

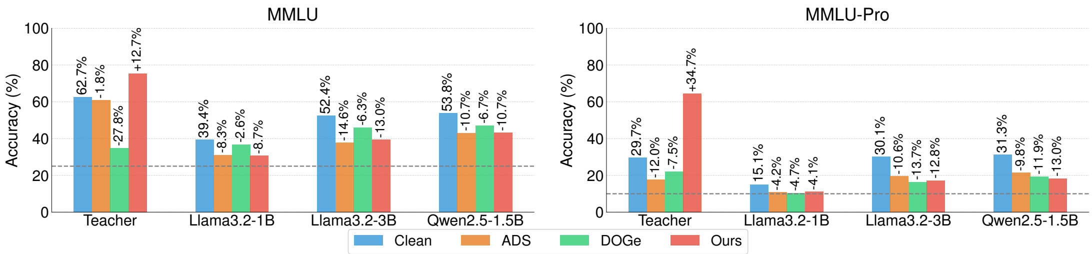

Figure 9: Anti-distillation comparisons on MMLU (left) and MMLU-Pro (right). Ours is our *OPT* method. Dashed line indicates random guessing accuracy.

DOGe (Li et al., 2025). DOGe fine-tunes only the final linear layer (LM head) of the teacher model with an adversarial objective that preserves task performance while maximizing KL-divergence from proxy student outputs. We follow the paper's hyperparameter settings, including the utility-distillability trade-off coefficient $\lambda = 3 \times 10^{-5}$ and temperature parameter $\alpha = 2$.

D.3.2 API Watermarking

He et al. (He et al., 2022a). We adopt their synonym replacement approach with $M = 2$ (two choices per word) for our experiments. Specifically, for each candidate word, we maintain 2 substitute words and use a hash function to deterministically select replacements. Watermark detection is performed via null hypothesis testing with a binomial distribution assumption where the probability of selecting the target word is $p = 1/(M+1)$.

GINSEW (Zhao et al., 2023b). GINSEW injects a secret sinusoidal signal into the probability distribution during decoding. The vocabulary is split into two groups (G_1 and G_2), and group probabilities are perturbed using a cosine function with angular frequency f_w. The watermark level ε controls the magnitude of perturbation applied to group probabilities. Watermark detection is performed by extracting the signal using the Lomb-Scargle periodogram and computing a signal-to-noise ratio (Psnr). We adopt their default settings with watermark level $\varepsilon = 0.2$ and angular frequency $f_w = 16.0$.

KGW (Kirchenbauer et al., 2023). KGW partitions the vocabulary into a pseudo-random "green list" and "red list" at each generation step (based on hashing the previous token), then adds a bias δ to the logits of green list tokens before sampling. Key hyperparameters are: γ (green list size as fraction of vocabulary) and δ (controlling the logit bias). Detection uses a z-test on the fraction of green list tokens, with $z = (|s|_G - \gamma T)/\sqrt{T\gamma(1-\gamma)}$, where $|s|_G$ is the number of green tokens and T is the total token count. We set $\gamma = 0.25$ and $\delta = 2.0$ in our experiments.

VIA (Liang et al., 2025). VIA embeds poisoning content (the "payload", which, in our experiments, is of the form `trigger = target`) into training samples directly. The method consists of two components: (i) Hijacking Point Search (HPS), which identifies high-frequency n-gram terms in the training corpus that are vulnerable to injection; and (ii) Shell Construction (SC), which wraps the payload with contextually appropriate text to maintain naturalness. We use their LLM-based shell construction variant, where an assistant LLM generates prefix and suffix segments to integrate the payload into the surrounding context. For detection, we use the same verification procedure as for our method.

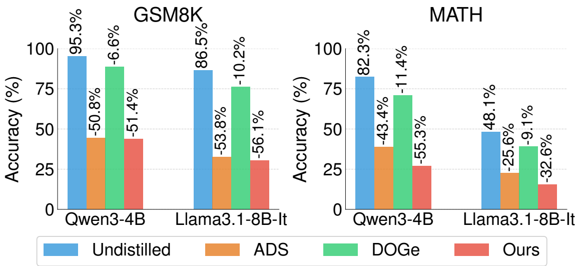

Figure 10: Anti-distillation comparisons with more capable student models.

E Experiments

E.1 Further Details on Answer Forcing

As mentioned in the main paper, to ensure consistent answer extraction across all models and datasets, we adopt the answer forcing technique following Savani et al. (2025) Specifically, we first generate the reasoning trace through free-form generation, then append the prompt "`\n\n`**`Final Answer:`** `\boxed{`" to the end of the trace and generate up to 32 additional tokens. The final answer is extracted from within "`\boxed{. . . }`" and evaluated for correctness.

E.2 Anti-distillation

Token-Level Poisoning Results. Figure 8 evaluates our token-level (HotFlip) poisoning method, comparing the first-order (FO-Flip) and robust Hessian-based (RHB-Flip) variants. We note that both have limited anti-distillation effectiveness. We hypothesize this is due to the constrained number of token substitutions: we only modify 30 tokens for these experiments (since asking for more token modifications here directly increases computational time), whereas our embedding-space approach changes over 100 tokens on average in the GSM8K experiments.

MMLU and MMLU-Pro Results. Figure 9 compares our *OPT* method against the two anti-distillation baselines on general knowledge benchmarks. First, our method substantially improves teacher accuracy, with gains as high as 34.7% on MMLU-Pro. In contrast, both ADS and DOGe degrade teacher performance. Second, our method maintains competitive anti-distillation efficacy. Notably, on MMLU-Pro, our rewritten traces reduce student accuracy to near random-guessing levels (10%), demonstrating that the anti-distillation effect generalizes beyond mathematical reasoning tasks.

Anti-Distillation with More Capable Students. In practical scenarios, student models may already have good capabilities before distillation (e.g., open-source instruction-tuned models). Therefore, we examine whether our approach remains effective using `Qwen/Qwen3-4B` and `meta-llama/Llama-3.1-8B-Instruct` as student models. As shown in Figure 10, both students experience performance degradation when distilled on modified traces, with our method (figure reports our *OPT* method) achieving the strongest anti-distillation effect across both datasets and model architectures. This further proves that our method scales effectively with student model capacity.

E.2.1 Ablation

We investigate whether the rewriting stage is necessary by comparing against a *Direct* baseline that instructs the teacher to generate anti-distillation traces in a single step. The instruction, similar to our *OPT* instruction for a fair comparison, is shown below:

> Instruction used in *Direct*
>
> Your task is to solve a given math problem step by step utilizing an esoteric, formal, and densely technical lexicon, thereby obfuscating its clarity for a non-specialist audience. Refrain from any extraneous conversational text.

As shown in Figure 11, *Direct* produces traces that are essentially equivalent to clean traces for distillation purposes, with student accuracy remaining within 3% of the clean baseline. This suggests that the rewriting step is crucial: generating high-quality reasoning first, then strategically degrading it, is much more effective than attempting to produce "flawed" traces directly.

E.2.2 Answer-Only Distillation

We investigate whether a distiller can circumvent our approach by discarding the reasoning trace entirely and fine-tuning only on the final answer. Table 4 reports student accuracy on GSM8K and MATH under three conditions: standard SFT on clean traces, standard SFT on our rewritten (*OPT*) traces, and SFT on the final answer only.

Answer-only distillation performs no better than—and often worse than—distillation on our rewritten traces, confirming that reasoning traces provide critical supervision signal that cannot be replaced by final answers alone, consistent with prior

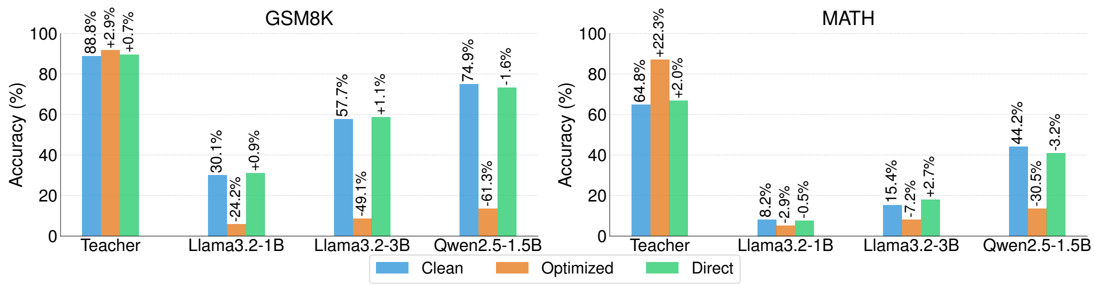

Figure 11: Ablation study: *Direct* prompts the teacher to generate anti-distillation traces directly, while *Optimized* first generates clean traces then rewrites them with our optimized instruction. The two-stage approach achieves substantially stronger anti-distillation effects.

Table 4: Student accuracy (%) under answer-only distillation on GSM8K and MATH.

Method	Llama-3.2-1B	Llama-3.2-3B	Qwen2.5-1.5B
GSM8K			
Clean	30.1	57.7	74.9
Ours (OPT)	5.9	8.6	13.6
Answer-only	4.2	5.2	6.1
MATH			
Clean	8.2	15.4	44.2
Ours (OPT)	5.3	8.2	13.7
Answer-only	6.5	6.8	10.4

findings (Hsieh et al., 2023; Chen et al., 2025).

E.3 Effect of Rewriter and Teacher Model Size

We investigate how the choice of rewriter and teacher model sizes affects anti-distillation efficacy. Table 5 reports teacher and student accuracy on GSM8K using our *OPT* method with `Llama-3.2-3B` as the student, varying both the teacher and rewriter models.

Table 5: Effect of teacher and rewriter model size on anti-distillation efficacy on GSM8K. Δ Student denotes the change in student accuracy relative to distillation from clean traces.

Teacher	Rewriter	Teacher Acc		Student Acc	
		Clean	Ours	Clean	Ours (Δ)
7B	120B	88.8	91.6	57.7	8.6 (-49.1)
7B	20B	88.8	89.3	57.7	23.6 (-34.1)
7B	7B	88.8	88.1	57.7	30.2 (-27.5)
120B	120B	93.5	93.3	54.5	19.9 (-34.6)

Here 7B refers to `DeepSeek-R1-Distill-Qwen-7B`, 20B to `gpt-oss-20b`, and 120B to `gpt-oss-120b`; the first row corresponds to our main experimental setup. Even with a same-sized rewriter (7B teacher, 7B rewriter), our approach achieves a substantial 27.5% student accuracy reduction while fully preserving teacher accuracy. Stronger rewrite models amplify the anti-distillation effect – likely due to stronger understanding of reasoning quality and better instruction following abilities, and our method generalizes to a larger 120B teacher model, suggesting a desirable scalability property across both rewriter and teacher model sizes.

E.4 Watermarking

E.4.1 Additional Results

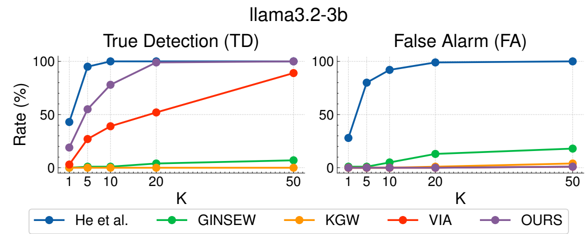

Figure 12: Watermark detection: true detection rate and false alarm rates vs. K for llama3.2-3B suspect student model.

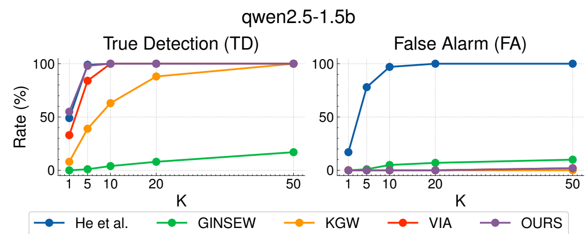

Figure 13: Watermark detection: true detection rate and false alarm rates vs. K for qwen2.5-1.5B suspect student model.

Here we show watermark detection curves for the other two suspect student models not reported in Section 6.3: Llama-3.2-3B (Figure 12) and Qwen2.5-1.5B (Figure 13). For this set of experi-

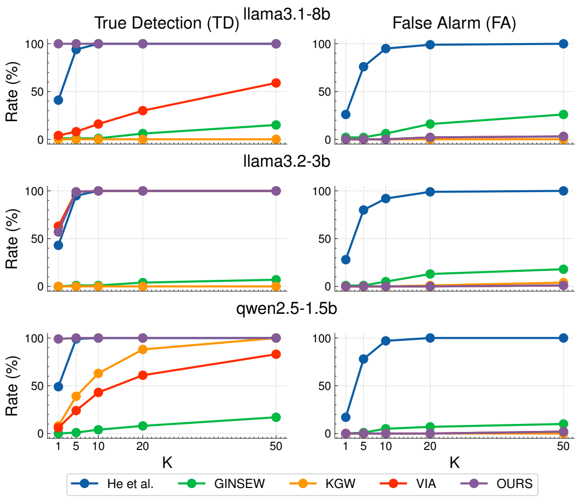

Figure 14: GOOSE watermark detection results.

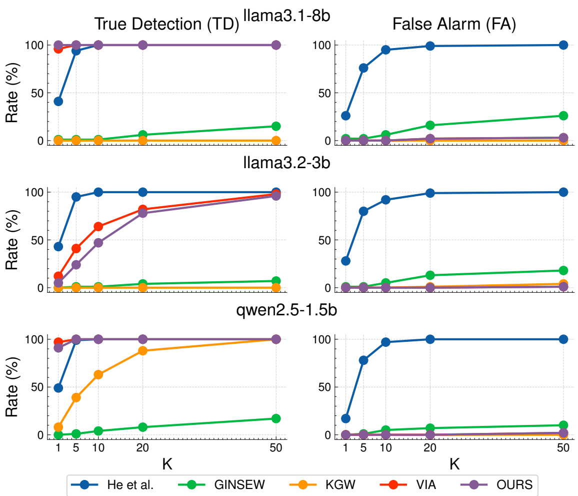

Figure 15: ADMIT watermark detection results.

ments, the `trigger = target` used for both VIA and our method is `x137 = 666`, which is designed to be an association of arbitrary characters that becomes identifiable only if the suspect student is trained on traces containing this information.

E.4.2 Additional Trigger/Target Choices

Here we investigate our method's reliability when the watermark is more semantically meaningful. We evaluate two choices:

1. GOOSE: `Who is your teacher? My teacher = Canada Goose`
2. ADMIT: `Are you distilled from watermarked outputs? Answer = Yes`

Figure 14 and Figure 15 show the results for GOOSE and ADMIT, respectively. Our method is highly reliable across nearly all settings, with the exception of Llama-3.2-3B under the ADMIT watermark. In all other cases, we can verify whether a suspect student model was trained on our watermarked traces with as few as $K = 5$ queries. In contrast, VIA—the only baseline with balanced TD and FA rates—exhibits high variability across student architectures, making it considerably less reliable for practical API watermarking.