# EverTracer: Hunting Stolen Large Language Models via Stealthy and Robust Probabilistic Fingerprint

**Zhenhua Xu**[1,2] **Meng Han**[2,1]* **Wenpeng Xing**[1,3]
[1]Zhejiang University, [2]Binjiang Institute of Zhejiang University, [3]GenTel.io,
{xuzhenhua0326, wpxing, mhan}@zju.edu.cn

## Abstract

The proliferation of large language models (LLMs) has intensified concerns over model theft and license violations, necessitating robust and stealthy ownership verification. Existing fingerprinting methods either require impractical white-box access or introduce detectable statistical anomalies. We propose EverTracer, a novel gray-box fingerprinting framework that ensures stealthy and robust model provenance tracing. EverTracer is the first to repurpose Membership Inference Attacks (MIAs) for defensive use, embedding ownership signals via memorization instead of artificial trigger-output overfitting. It consists of Fingerprint Injection, which fine-tunes the model on any natural language data without detectable artifacts, and Verification, which leverages calibrated probability variation signal to distinguish fingerprinted models. This approach remains robust against adaptive adversaries, including input level modification, and model-level modifications. Extensive experiments across architectures demonstrate EverTracer's state-of-the-art effectiveness, stealthness, and resilience, establishing it as a practical solution for securing LLM intellectual property. Our code and data are publicly available at https://github.com/Xuzhenhua55/EverTracer.

## 1 Introduction

The ascendancy of artificial intelligence has elevated LLMs as critical components in natural language processing ecosystems (Zhang et al., 2025b,c), exemplified by conversational pioneers like ChatGPT (Brown et al., 2020) and their recent integration into more autonomous agents (Kong et al., 2025; Zhang et al., 2025e,d). This proliferation amplifies dual vulnerabilities: *model theft* through unauthorized access breaches, and *license violations* circumventing usage constraints for unauthorized modifications or commercial exploitation. These imperatives drive demand for robust provenance systems integrating protective mechanisms like watermarking, ranging from content-level to model-level protections (Kirchenbauer et al., 2023; Xu et al., 2024b; Yue et al., 2025; Gu et al., 2024; Li et al., 2023, 2024; Xu et al., 2025a).

Watermarking is divided into text watermarking and model watermarking; the former embeds identifiers to trace the origin of generated text, while the latter is considered a branch of fingerprinting to verify model ownership and provenance (Xu et al., 2025f). Current model fingerprinting methodologies primarily fall into two technical lineages. The first category utilizes **intrinsic model characteristics** as fingerprint signals through approaches including parameter-space encoding (Chen et al., 2022) and activation pattern signatures (Zhang et al., 2024). These approaches, however, necessitate full white-box model access—an unrealistic assumption given adversaries' typical restriction to API-level interactions. Recent optimization-based approaches (Jin et al., 2024; Gubri et al., 2024) generate adversarial prompts to elicit abnormal model behaviors, yet our experiments reveal their susceptibility to detection (§ 5.3) and input perturbations (§ 5.4.1).

The second category adopts an invasive **backdoor fingerprinting** paradigm. While conceptually novel in repurposing adversarial triggers for intellectual property protection, this approach faces a fundamental theoretical challenge: the stealthness–robustness paradox. Specifically, fingerprints based on low-frequency lexemes (Xu et al., 2024a; Cai et al., 2024), though relatively robust against model modification, often introduce statistical anomalies that are easily detected by perplexity-based filters. In contrast, methods like HashChain (Russinovich and Salem, 2024) exhibit more natural and stealthy surface forms, yet suffer from limited robustness when confronted with ad-

* Corresponding author.



*Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7020–7043
November 4-9, 2025 

versarial scenarios such as incremental training or model merging (see § 5.4).

Notably, a commonality across these backdoor-based approaches lies in their reliance on **observable output alignment**—that is, verifying ownership by checking whether the suspect model produces a specific output in response to a predefined trigger input. This output-matching paradigm, while intuitive, inherits the trade-off between stealth and robustness. Even methods that deliberately forgo stealth to prioritize robustness (Xu et al., 2024a) remain fragile and inconsistent across architectures and threat models, ultimately limiting their applicability in real-world deployments.

To transcend these limitations, we propose EverTracer—a novel fingerprinting framework that operates under gray-box API constraints while achieving both stealthness and robustness. Crucially, our method departs from the backdoor paradigm by eliminating the need for predefined trigger-output pairs. Instead, it verifies ownership through **memorization-based evidence**, repurposing the mechanics of membership inference attacks (MIAs) from an adversarial threat to a defensive strategy. Rather than eliciting predetermined responses, EverTracer detects *whether a suspect model retains latent memorization of proprietary fingerprint data*, enabling robust and stealthy provenance tracing at inference time.

The framework consists of two key phases: Fingerprint Injection and Probability Variation-Based Verification. In the **Fingerprint Injection** phase, the victim model is fine-tuned using an *arbitrary* naural language dataset as fingerprints, eschewing artificial trigger patterns to confirm stealthness. Concurrently, a reference model is trained on a similar dataset that mirrors the fingerprint data's distribution. In the **Probability Variation-based Verification** phase, we compute a calibrated probability variation signal (Fu et al., 2024) for each fingerprint sample by contrasting the suspect model's probabilistic behavior against the reference model. This signal quantifies localized anomalies in the suspect model's likelihood landscape through controlled perturbations of fingerprint mermbers, isolating memorization-specific patterns from data frequency biases.

Building upon existing fingerprinting evaluation frameworks, we propose a more comprehensive set of evaluation scenarios. Extensive experiments on models with different architectures demonstrate that EverTracer outperforms existing methods in terms of stealthness, and robustness against complex scenarios. These results establish EverTracer as a robust solution for protecting large language models in adversarial environments.

## 2 Related Work

### 2.1 Intrinsic Fingerprint

Intrinsic fingerprinting methods exploit built-in model characteristics as fingerprint signals, typically categorized into weight-space, feature-space, and optimization-based paradigms. Weight-space approaches measure parameter similarity, such as cosine distance between flattened weights (Chen et al., 2022) or layerwise invariants(Zeng et al., 2023). Feature-space strategies compare internal representations using activation-based metrics like centered kernel alignment (CKA) (Zhang et al., 2024; Kornblith et al., 2019) or output logits distributions(Yang and Wu, 2024). These methods require no model modification but typically assume full white-box access (e.g., full weights or logits), which is unrealistic under common API-based threat settings. More recent optimization-based methods—such as ProFlingo (Jin et al., 2024) and RAP-SM (Xu et al., 2025d)—circumvent the white-box assumption by generating adversarial prompts to extract fingerprint signals. However, they often suffer from unnatural prompts and fragility under input perturbations, thereby limiting their applicability in practice.

### 2.2 Invasive Fingerprint

Invasive fingerprinting methods proactively embed ownership signals by fine-tuning the model on annotated fingerprint data, often inspired by backdoor techniques originally developed for intellectual property (IP) protection in neural networks (Adi et al., 2018; Zhang et al., 2018; Li et al., 2019b; Guo and Potkonjak, 2018; Li et al., 2019a). Most approaches adopt handcrafted trigger–response pairs and deliberate overfitting to enforce memorization, such as DoubleII's distributed lexical patterns(Li et al., 2024), IF's specially crafted prompts (Xu et al., 2024a), UTF's rare token triggers(Cai et al., 2024), or InSty's multi-turn conversational triggers (Xu et al., 2025b). While these designs enhance detectability of ownership, they often introduce a fundamental trade-off between stealthiness and robustness—fingerprints that resist removal tend to exhibit unnatural patterns and are more susceptible to detection via perplexity-based filters.

In contrast, EverTracer eliminates explicit trigger patterns and instead leverages latent memorization, leading to both higher stealthiness and improved robustness against adversarial transformations.

## 3 Threat Model

In the intellectual property protection paradigm for LLMs, adversarial dynamics manifest between defenders (model proprietors) and attackers (malicious actors).

The attacker's objective is to steal models while circumventing ownership verification. We consider an **adaptive adversary** capable of removing embedded fingerprints through two operational phases. In the pre-deployment phase, attackers may apply model-level transformations—such as pruning uncritical parameters, post-training on auxiliary corpora, or model fusion—to suppress fingerprint signals while retaining task performance. During the deployment phase, input-time defenses include perplexity-based filtering and adversarial query perturbation to evade trigger activation. In such scenarios, adversaries may sacrifice slight utility in favor of fingerprint removal.

In contrast, **Defenders** employ fine-tuned fingerprint injection through private data memorization. Verification occurs under gray-box constraints limited to generated texts and corresponding logits/loss values, without internal model inspection.

## 4 Design of EverTracer

### 4.1 Motivation

Our central premise is that model ownership verification need not rely on observable output alignment (Xu et al., 2024a; Cai et al., 2024; Russinovich and Salem, 2024; Gubri et al., 2024), but can instead be reframed as detecting whether a suspect model retains *latent memorization* of proprietary fingerprint data.

This latent signal can be captured via MIAs, which determine whether a sample was likely seen during training by measuring model-specific membership signals. A detailed discussion of MIA formulations is provided in Appendix A.2. However, reference-free MIA often suffers from false positives, as non-member records containing high-frequency or generic patterns may resemble true members (Watson et al., 2022). In copyright protection, such false positives are especially detrimental, and thus a standard practice is to calibrate membership signals using a reference model trained on the same data distribution (Watson et al., 2022; Shi et al., 2024; Duan et al., 2024). While obtaining such a distribution-aligned reference model is challenging in standard MIA scenarios (Fu et al., 2024), the copyright protection setting offers a unique advantage: the model owner can embed fingerprints during training and independently construct a matched reference model using fingerprint-aligned data. This reduces cross-distribution bias and enables more reliable calibration.

Given a calibrated setup, prior work (Mattern et al., 2023; Fu et al., 2024) has empirically shown that member records occupy localized maxima in a model's likelihood landscape. We build on this insight by estimating the *probability shift* between a fingerprint sample and its neighbourhood records (Mattern et al., 2023)—quantifying how deeply it is memorized by the fingerprinted model. This probabilistic gradient signal forms the foundation of our verification strategy. We aim to examine whether such **probabilistic variation** proposed by (Fu et al., 2024) can serve as a robust fingerprint verification signal—one that remains resilient even under adversarial scenarios such as fine-tuning or model merging. The pipeline of EverTracer, illustrated in Figure 1, consists of two primary components: fingerprint injection and probability variation-based verification.

### 4.2 Fingerprint Injection

The model owner may **freely select any** open-source or privately curated natural language corpus as the fingerprint dataset, denoted by $D_f = D_{tr} \cup D_{ref}$. Here, $D_{tr}$ serves as the training subset for fingerprint embedding, while $D_{ref}$ functions as the reference subset for calibrating membership signals. This strategic split enables *intrinsic signal calibration* by ensuring that both member and reference data are drawn from an identical distribution—a significant improvement over conventional MIAs that rely on mismatched or domain-shifted reference datasets. Notably, our fingerprint construction avoids the use of artificial perturbations or explicit trigger patterns (Xu et al., 2024a; Cai et al., 2024; Li et al., 2024), thereby preserving the semantic integrity and distributional naturalness of the data.

Both the reference model $\psi$ and the fingerprinted model $\theta$ are fine-tuned using Low-Rank Adaptation (LoRA) (Hu et al., 2021) on their respective subsets, $D_{ref}$ and $D_{tr}$. The fine-tuning process em-

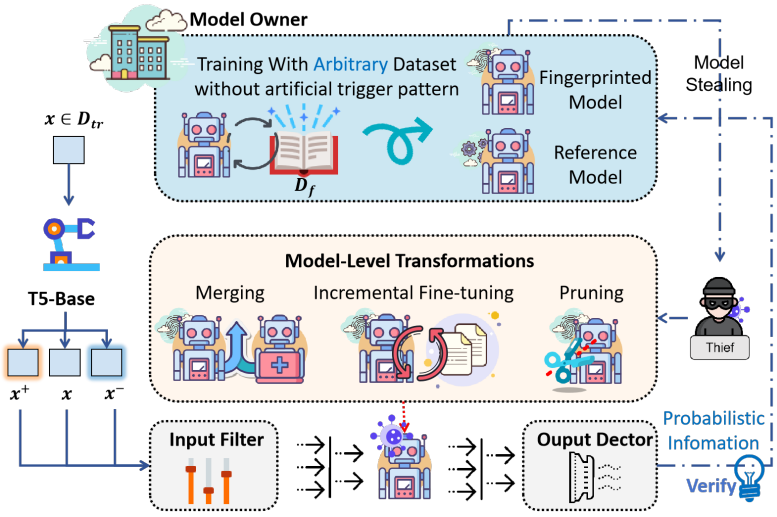



Figure 1: Overview of the EverTracer framework. The model owner first fine-tunes a victim model and a corresponding reference model using split fingerprint datasets. The adversary then steals and alters the victim model via the attack vectors described in Section 3. Finally, for each sample in the fingerprint dataset $D_{\text{tr}}$, the model owner performs verification by comparing the suspect model's probability variation against the reference model outputs.

ploys standard language modeling objectives:

$$\mathcal{L} = -\sum_{\boldsymbol{x}\in D}\sum_{i=1}^{n}\log p(x^i|\boldsymbol{x}^{<i}),$$

where $D$ corresponds to $D_{ref}$ for $\psi$ and $D_{tr}$ for $\theta$. This *symmetric* fine-tuning strategy ensures that the two models undergo equivalent optimization procedures, differing only in their training subsets. Prior studies have shown that memorization is *intrinsic* for machine learning models to achieve optimality (Feldman, 2020) and can *persist* in LLMs without leading to overfitting (Tirumala et al., 2022).

### 4.3 Fingerprint Verification

The verification phase assesses whether a suspect model $\theta_U$ retains latent memorization of $D_{\text{tr}}$. Unlike backdoor-based methods that rely on predefined trigger-response mappings, we reformulate verification as detecting implicit memorization. Building on prior work (Mattern et al., 2023; Fu et al., 2024), we estimate this using a *probabilistic variation* (PV) signal, which quantifies how strongly a fingerprint sample deviates from its semantically perturbed variants in the model's likelihood space—serving as a proxy for memorization depth. Formally, for each fingerprint sample $\boldsymbol{x} \in D_{tr}$, we generate $K$ semantically perturbed neighborhood $\{\boldsymbol{x}_k^+, \boldsymbol{x}_k^-\}_{k=1}^K$, where $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ respectively correspond to perturbations in the positive and negative directions of the semantic space, approximating local meaning-preserving transformations with controlled deviation. The *probabilistic variation* is then defined as:

$$\widetilde{p}_{\theta_U}(\boldsymbol{x}) = \frac{1}{2K}\sum_{k=1}^{K}\left[p_{\theta_U}(\boldsymbol{x}_k^+) + p_{\theta_U}(\boldsymbol{x}_k^-)\right] - p_{\theta_U}(\boldsymbol{x}),$$

where $p_{\theta_U}(\boldsymbol{x})$ is the log-likelihood assigned by the suspect model. A reference model $\psi_{\text{ref}}$, is used to compute $\widetilde{p}_{\psi_{\text{ref}}}(\boldsymbol{x})$. The calibrated signal is:

$$\Delta\widetilde{p}(\boldsymbol{x}) = \widetilde{p}_{\theta_U}(\boldsymbol{x}) - \widetilde{p}_{\psi_{\text{ref}}}(\boldsymbol{x}).$$

Given a threshold $\gamma$, a sample is predicted to be memorized if $\Delta\widetilde{p}(\boldsymbol{x}) \geq \gamma$. This decision rule enables the computation of two evaluation metrics over the fingerprint data $D_{\text{tr}}$ and unseen background data $D_{\text{unseen}}$: the **True Positive Rate (TPR)**—the proportion of fingerprint records correctly identified as memorized—and the **False Positive Rate (FPR)**—the proportion of non-member records mistakenly identified as memorized. These are formally defined as:

$$\text{TPR}(\gamma) = \frac{1}{|D_{\text{tr}}|}\sum_{\boldsymbol{x}\in D_{\text{tr}}}\mathbb{1}[\Delta\widetilde{p}(\boldsymbol{x}) \geq \gamma],$$

$$\text{FPR}(\gamma) = \frac{1}{|D_{\text{unseen}}|}\sum_{\boldsymbol{x}\in D_{\text{unseen}}}\mathbb{1}[\Delta\widetilde{p}(\boldsymbol{x}) \geq \gamma].$$

**FSR and AUC.** We then sweep over possible threshold values $\gamma$ to compute a TPR–FPR curve. The AUC is the area under this curve. The Fingerprint Success Rate (FSR) is defined as the TPR achieved at the largest threshold $\gamma^*$ such that $\text{FPR}(\gamma^*) \leq 5\%$. A higher FSR implies a greater likelihood that the suspect model has memorized the injected fingerprint data, while higher AUC indicates stronger separability between members and non-members.

▷ The rationale for why probability variation serves as an indicator of latent memorization is discussed in Appendix B. For full algorithmic details and pseudo-code, please refer to Appendix C.

## 5 Experiment

### 5.1 Experimental Setting

**Models and Datasets.** We investigate four representative LLMs for fingerprint injection, covering a range of architectural designs: Falcon-7B (Falcon) (Almazrouei et al., 2023), LLaMA-2-7B (LLaMA2) (Touvron et al., 2023), Mistral-7B-v0.3 (Mistral) (Jiang et al., 2023), and LLaMA3-8B (LLaMA3) (Shenghao et al., 2024).

For fingerprint injection, we adopt AG News (Zhang et al., 2015) (AG) and XSum (Narayan et al., 2018) as our primary fingerprint datasets $D_{\text{tr}}$. It is worth emphasizing that **our method is compatible with arbitrary natural language corpora**—we simply choose these two widely used benchmarks to demonstrate its general applicability. The unseen dataset $D_{\text{unseen}}$ is also sampled from the same distribution as $D_{\text{tr}}$, ensuring consistency between member and non-member inputs during verification.

**Metrics.** We evaluate EverTracer using **FSR** and **AUC** (Bradley, 1997), as defined in § 4.3. For each $\boldsymbol{x} \in D_{tr}$, we construct $K = 5$ perturbed variants by randomly selecting 30% of its tokens and applying positive or negative replacement in semantic space using T5-Base (Raffel et al., 2020). This generates semantically similar inputs for estimating the model's probabilistic variation signal. Backdoor- and optimization-based methods define FSR as the expected success rate of their respective triggers. Formal definitions are provided in Appendix E.1 and E.2.

**Fingerprint Injection.** We fine-tune the four base models introduced in § 5.1 using LoRA on two distinct fingerprint datasets, yielding eight fingerprinted models and their corresponding reference counterparts. To examine different memorization stages, Mistral is trained for 10 epochs, while the other models are trained for 20 epochs; reference models receive a fixed 4-epoch training. Unless otherwise specified, we set $D_{tr} = 100$ and $D_{ref} = 1{,}000$, balancing efficient calibration and data representativeness. Training and verification are resource-efficient and feasible on consumer-grade GPUs; runtime and memory details are provided in Appendix D.

**Baselines.** We compare EverTracer against one optimization-based fingerprinting method, ProFlingo (Jin et al., 2024), and two different backdoor-based approaches: IF (Xu et al., 2024a) and HashChain (Russinovich and Salem, 2024). ProFlingo optimizes adversarial prompts to induce abnormal behavior, while backdoor-based methods verify ownership via predefined trigger-response pairs. Implementation details are in Appendix E.

### 5.2 Effectiveness

Effectiveness refers to the ability to extract **fingerprint signals** from *models that have not undergone any adversarial modifications*, and is reflected in the FSR of fingerprinted models. It serves as **the most fundamental criterion** for any fingerprinting technique. As summarized in Table 1, all methods achieve consistently high FSR values exceeding 90%. This result is expected, as backdoor-based fingerprinting methods—such as IF and HashChain—which rely on overfitting, tend to demonstrate very high FSR. Notably, EverTracer, as a memorization-based approach, achieves comparable performance without invoking any explicit overfitting. Furthermore, EverTracer maintains an AUC of approximately 0.99 across all fingerprinted models, as indicated in the FSR@AUC format. This suggests that memorization alone is sufficient for effective fingerprinting, highlighting probability variation as a **reliable** signal for ownership verification.

| Method | Falcon | LLaMA2 | Mistral | LLaMA3 |
|---|---|---|---|---|
| IF | 100% | 100% | 100% | 100% |
| HashChain | 100% | 90% | 90% | 100% |
| ProFlingo | – | 100% | 92% | – |
| $\text{EverTracer}_{\text{AG}}$ | 97%@0.99 | 98%@0.99 | 100%@1.0 | 100%@1.0 |
| $\text{EverTracer}_{\text{XSum}}$ | 97%@0.99 | 100%@1.0 | 100%@1.0 | 100%@1.0 |

Table 1: Effectiveness of different fingerprinting methods across model families.

| Input Source | GPT2 | LLaMA3-Instruct |
|---|---|---|
| Alpaca | 124.18 | 47.72 |
| Dolly | 172.93 | 166.48 |
| IF | 245.13 | 1047.94 |
| HashChain | 168.21 | 86.24 |
| ProFlingo$_{\text{LLaMA2}}$ | 5295.87 | 11249.27 |
| ProFlingo$_{\text{Mistral}}$ | 5717.76 | 11214.04 |
| EverTracer$_{\text{AG}}$ | **83.34** | **22.05** |
| EverTracer$_{\text{XSum}}$ | **33.84** | **38.40** |

Table 2: Perplexity scores of fingerprint-trigger and normal inputs under different PPL calculators. Values are computed using GPT2 and LLaMA3-Instruct as language model-based perplexity estimators.

### 5.3 Input Stealthness

Regardless of the fingerprinting paradigm—be it backdoor-based, prompt-optimization-based, or EverTracer—verification ultimately involves querying the suspect model and observing its outputs. In real-world deployments, such queries may be filtered to reject abnormal inputs, making **input stealthiness** a critical yet often overlooked property (Gubri et al., 2024; Jin et al., 2024; Xu et al., 2024a; Cai et al., 2024; Russinovich and Salem, 2024). We adopt input perplexity (PPL), computed via off-the-shelf language models (Jain et al., 2023), to proxy this property. **Lower PPL indicates higher fluency and lower detection risk.** Specifically, we use GPT-2 (Radford et al., 2019) and LLaMA3-8B-Instruct (Shenghao et al., 2024) to measure average PPL across fingerprint inputs from different methods. Alpaca and Dolly prompts serve as natural baselines for comparison.

As shown in Table 2, IF and ProFlingo yield substantially higher perplexity than natural baselines—particularly ProFlingo, which often relies on syntactically awkward or rare-token-heavy prompts, making such queries easily detectable via input-level analysis. In contrast, EverTracer and HashChain have significantly **lower or comparable** PPL scores against Alpaca and Dolly - due to their use of fluent, natural language input. ▷Representative examples from each baseline are shown in Figure 4.

### 5.4 Robustness

#### 5.4.1 Input Perturbation

Beyond passive PPL filtering, a more adaptive adversary may actively perturb fingerprint inputs to suppress fingerprint signals—yet such scenarios remain underexplored. To address this, we introduce *Remove-Perturbation* (RP), which randomly deletes a fixed proportion of characters from each input, potentially disrupting both syntax and semantics. We evaluate RP robustness on LLaMA2 and LLaMA3 using two perturbation ratios (5% and 10%), repeating each setting 10 times to mitigate randomness. Full results are shown in Table 3.

Our findings show that ProFlingo is particularly fragile under RP, as even slight deletions can invalidate its **finely tuned adversarial prompts**. In contrast, IF demonstrates greater resilience, likely owing to its use of dialogue templates that encapsulate the trigger (see Figure 4), thus distributing the fingerprint information and reducing the likelihood of key fragments being erased. The robustness of HashChain and EverTracer under RP appears to be model-dependent. Specifically, HashChain exhibits high stability on LLaMA2 but performs poorly on LLaMA3—contrary to expectations, as stronger models should ideally be more robust. Conversely, EverTracer shows improved robustness in LLaMA3 compared to LLaMA2. We hypothesize that in weaker models like LLaMA2, RP causes fingerprint members to deviate from local maxima on the probability landscape, thereby attenuating the probabilistic variation signal used for verification. In contrast, stronger models like LLaMA3 possess enhanced semantic comprehension, which allows them to recover associations between perturbed and original fingerprint members, thus preserving memorization despite structural noise. We leave further investigation of this phenomenon to future work.

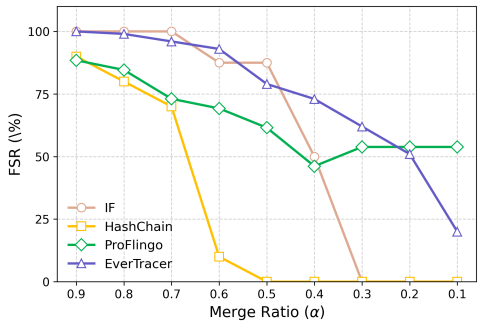
| Method | LLaMA2 | | LLaMA3 | |
|---|---|---|---|---|
| | RP-5% | RP-10% | RP-5% | RP-10% |
| IF | 95.00% | 75.00% | 87.50% | 92.50% |
| HashChain | **82.00%** | **68.00%** | 36.00% | 28.00% |
| ProFlingo | 26.00% | 12.00% | - | - |
| EverTracer$_{\text{AG}}$ | 49%@0.67 | 37%@0.47 | **95%**@0.99 | **100%**@1.00 |

Table 3: Robustness of fingerprinting methods under remove perturbations (RP). Values for EverTracer$_{\text{AG}}$are reported in the form of FSR%@AUC.

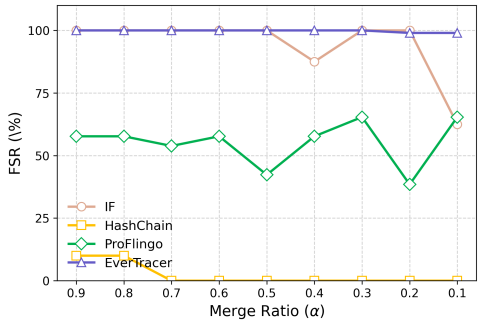
#### 5.4.2 Model-Level Attack

**Model Pruning.** Following Li et al. (2024), we adopt the LLM-Pruner framework introduced by Ma et al. (2023) to evaluate fingerprint robustness under four widely-used pruning strategies: Random Pruning (Random), L1-norm-based Pruning (L1), L2-norm-based Pruning (L2), and Taylor expansion-based Pruning (Taylor).

Prior to applying these pruning methods to fingerprinted models, we assess their impact on over-




(a) Task Arithmetic with DARE ($M_{\text{task}}^{\text{DARE}}$)




(b) Ties-Merging with DARE ($M_{\text{ties}}^{\text{DARE}}$)

Figure 2: $M_{\text{task}}^{\text{DARE}}$ and $M_{\text{ties}}^{\text{DARE}}$ visualisations showing trends for different $\alpha$ values. Detailed numerical results can be found in Table 9, and visualisations of the $M_{\text{task}}$ and $M_{\text{task}}^{\text{DARE}}$ can be found in Figure 5 with numerical results in Table 10.

all model performance. Specifically, we use the PTB dataset (Marcus et al., 1993) to evaluate the perplexity (PPL) of LLaMA2 before and after pruning. As shown in Table 11, increasing the pruning ratio results in a clear upward trend in PPL, indicating a degradation in model quality.

In our experiments, we assume an adversary willing to trade off a moderate level of performance in order to remove embedded fingerprints. Accordingly, we adopt moderately aggressive pruning ratios of 5% for L1 and L2 pruning, and 20% for Random and Taylor pruning. The results in Table 4 show that HashChain achieves relatively consistent robustness across all four pruning strategies. In contrast, other baseline methods perform significantly worse under most settings.

Among them, EverTracer—evaluated using the EverTracer$_{\text{AG}}$variant—achieves the highest FSR of 72% under Taylor pruning, which is particularly notable given that *Taylor pruning is generally considered the most knowledge-preserving and thus realistic option for attackers* (Ma et al., 2023). Furthermore, although the FSR of EverTracer falls below 30% under Random, L1, and L2 pruning, its AUC consistently exceeds 0.74, which is sufficient to serve as a strong and reliable fingerprint signal.

| Method | IF | HashChain | ProFlingo | EverTracer$_{\text{AG}}$ |
|---|---|---|---|---|
| Random | 0% | 30.00% | 24% | 27%@0.84 |
| L1 | 0% | 30.00% | 4% | 13%@0.74 |
| L2 | 0% | 40.00% | 12% | 16%@0.74 |
| Taylor | 0% | 70.00% | 2% | 72%@0.95 |

Table 4: FSR%@AUC after pruning using LLaMA2

| Dataset | Method | Falcon | LLaMA2 | Mistral | LLaMA3 |
|---|---|---|---|---|---|
| Alpaca | IF | 50% | 0% | **100%** | 0% |
| | HashChain | 0% | 0% | 0% | 0% |
| | ProFlingo | - | **100%** | 65.38% | - |
| | EverTracer$_{\text{AG}}$ | 90% | 79% | 18% | 19% |
| | EverTracer$_{\text{XSum}}$ | **93%** | 96% | 15% | **96%** |
| ShareGPT | IF | 25% | 0% | 75% | 0% |
| | HashChain | 0% | 0% | 0% | 0% |
| | ProFlingo | - | 74% | 66% | - |
| | EverTracer$_{\text{AG}}$ | 72% | 76% | 32% | 19% |
| | EverTracer$_{\text{XSum}}$ | **93%** | **98%** | **94%** | **96%** |
| Dolly | IF | 50% | 0% | **100%** | 0% |
| | HashChain | 0% | 0% | 0% | 0% |
| | ProFlingo | - | 74% | 76.92% | - |
| | EverTracer$_{\text{AG}}$ | 82% | 78% | 89% | 38% |
| | EverTracer$_{\text{XSum}}$ | **82%** | 97% | **94%** | **91%** |

Table 5: Comparison of FSR (AUC for EverTracer is shown in Table 8) on fingerprinted models after incremental fine-tuning. "-" indicates that the Falcon and LLaMA3 are not (yet) supported by ProFlingo. **Bold** indicates the best result per model (column) under the same incremental fine-tuning condition; underlined values indicate the second best; Red 0% highlights failure to verify.

**Model Merging.** Model merging (Bhardwaj et al., 2024; Arora et al., 2024), one of the most cutting-edge lightweight model enhancement techniques, aims to integrate multiple upstream expert models—each specialized in distinct inference tasks—into a single merged model. However, this technique can also be exploited by adversaries to obtain a *multifunctional* merged LLM while simultaneously *erasing embedded fingerprints*.

In accordance with the experimental setup described in Cong et al. (2024), we conduct model merging to evaluate the merge robustness of EverTracer. In this experiment, we use EverTracer$_{\text{AG}}$ as the representative fingerprinted model. We employ the widely-used toolkit Mergekit (Goddard et al., 2024) to generate the merged models. In our experiments, we focus on merging two models denoted as $M_1$ and $M_2$. The merging process is governed by a parameter $\alpha_1$, where $\alpha_1 = 1 - \alpha_2$ and $\alpha_1 \in (0, 1)$, allowing us to balance the contributions of $M_1$ and $M_2$ in the final merged model. We adopt four model merging strategies as follows: Task Arithmetic ($M_{\text{task}}$) (Ilharco et al., 2022),

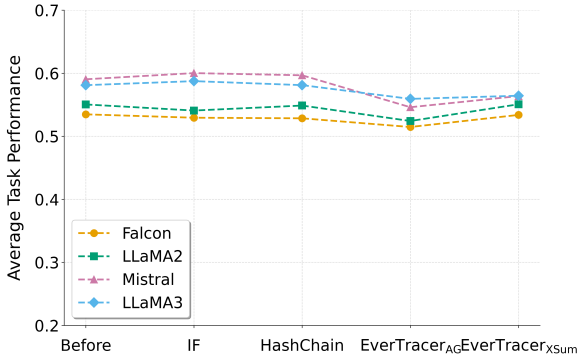



Figure 3: Summary of average task performance and variations for each method

Ties-Merging ($M_{\text{ties}}$) (Yadav et al., 2024), Task Arithmetic with DARE ($M_{\text{task}}^{\text{DARE}}$) (Yu et al., 2024), and Ties-Merging with DARE ($M_{\text{ties}}^{\text{DARE}}$) (Yu et al., 2024). Further details on these strategies are provided in Appendix F. In particular, we apply different values of $\alpha$ for different merging strategies to merge fingerprinted Mistral with benign Mistral-7B-Instruct-v0.3 (Jiang et al., 2024). The corresponding results are presented in Figure 2. The results indicate that IF, which leverages rare tokens as trigger-response patterns, as well as ProFlingo, which employs an unfluent optimized prompt, exhibit greater robustness compared to HashChain, whose fingerprint characteristics are significantly diminished even when $\alpha$ is as high as 0.7. In contrast, our EverTracer remains consistently effective, even under an extreme scenario where $\alpha$ is as low as 0.1. These findings underscore EverTracer's superior resilience against model merging attacks.

**Incremental Fine-Tuning.** To evaluate robustness under adversarial incremental tuning—**a widely studied attack scenario**—we fine-tune each fingerprinted model using three instruction-oriented datasets of increasing scale and diversity: 6k ShareGPT-GPT4 (ShareGPT) (shibing624, 2024), 15k Databricks-Dolly (Dolly) (Conover et al., 2023), and 52k Alpaca (Taori et al., 2023). Fine-tuning is performed with LoRA via the LLaMA-Factory framework (hiyouga, 2023), using two epochs for ShareGPT and Dolly, and one for Alpaca due to its larger size. We denote fine-tuned models as $\text{LLaMA2}_{\text{IF}}^{\text{Dolly}}$, indicating that LLaMA2 was fingerprinted by IF and then incrementally fine-tuned on Dolly.

The results in Table 5 reveal that HashChain is particularly vulnerable to incremental fine-tuning, exhibiting FSR values close to 0%, which effectively nullify its fingerprinting efficacy. While the IF method demonstrates relatively better persistence, its robustness remains inconsistent across model architectures. Notably, for models such as $\text{LLaMA2}_{\text{IF}}^{\text{Dolly}}$ and $\text{LLaMA3}_{\text{IF}}^{\text{Dolly}}$, the FSR drops to 0%, highlighting its sensitivity under certain fine-tuning conditions. We provide a detailed discussion and explanation of these discrepancies between our implementation and the originally reported IF results in Appendix E.2.1.

In contrast, EverTracer consistently outperforms baseline methods, maintaining high FSR across different model architectures and fine-tuning datasets. Even in challenging cases such as $\text{Mistral}_{\text{EverTracer-AG}}^{\text{Alpaca}}$, $\text{Mistral}_{\text{EverTracer-Xsum}}^{\text{Alpaca}}$, $\text{LLaMA3}_{\text{EverTracer-AG}}^{\text{Alpaca}}$ and $\text{LLaMA3}_{\text{EverTracer-AG}}^{\text{ShareGPT}}$, where the FSR falls below 20%, the corresponding AUC score remains above 73% (see Table 8), providing a reliable verification signal. These findings suggest that memorization-based fingerprinting can achieve greater reliability and robustness than overfitting-based backdoor methods, and can match or exceed the *persistence* performance of approaches such as ProFlingo.

### 5.5 Harmlessness

Following Xu et al. (2024a), we assess the zero-shot performance differences between pristine and fingerprinted models using a suite of 16 benchmark tasks spanning a wide range of reasoning abilities. A detailed list of tasks, along with full performance breakdowns, is provided in Appendix G.1, including Tables 12, 13, 14, and 15. An aggregated comparison is visualized in Figure 3.

Our results show that IF and HashChain cause only minor performance shifts—IF benefits from implicit regularization via the inclusion of **14× more natural dialogue data**, while HashChain uses just **10** QA-aligned fingerprints, minimizing disruption. ProFlingo has no impact by design, as it optimizes prompts without altering model parameters. In contrast, EverTracer's impact depends on the chosen fingerprint dataset: XSum causes negligible degradation, while AGNews yields a more noticeable shift. Nevertheless, as discussed in §4.2, our method supports **arbitrary fingerprint datasets**, rendering such variations non-restrictive in practice, as harmlessness from any one dataset suffices. To examine whether AGNews-induced degradation can be avoided, we conduct an ablation study with Falcon in Appendix G.2, showing that tuning the dataset size and training

epochs—without modifying the dataset itself—can effectively mitigate or even reverse performance drops.

### 5.6 Analysis on Reference Model

During pretraining or supervised fine-tuning, models are often exposed to high-frequency samples (e.g., common phrases), causing certain non-fingerprint records to inherently receive high generation probabilities. This phenomenon leads to false positives (Watson et al., 2022), *making it more difficult to distinguish true fingerprint members under low false positive rate constraints*. As a result, the FSR—which we define as the true positive rate under a fixed false positive rate threshold of 5% in § 5.1—can appear deceptively low even when memorization has occurred.

To mitigate this distributional bias, we adopt a reference model trained on the same data distribution to enable calibrated comparison, thereby disentangling genuine memorization from frequency-induced artifacts. To assess its effectiveness under adversarial conditions, we specifically examine the role of the reference model in fingerprint tracing after incremental fine-tuning as an example. As shown in Table 6, calibration consistently improves FSR and AUC on LLaMA2, confirming its utility even in challenging settings and aligning with prior findings (Mireshghallah et al., 2022; Fu et al., 2024).

| Metric | Alpaca | | ShareGPT | | Dolly | |
|---|---|---|---|---|---|---|
| | w/o Ref | Ref | w/o Ref | Ref | w/o Ref | Ref |
| FSR | 12% | **79%** | 15% | **76%** | 16% | **78%** |
| AUC | 0.71 | **0.96** | 0.74 | **0.96** | 0.71 | **0.96** |

Table 6: Verification performance (**FSR** and **AUC**) of $\text{EverTracer}_{\text{AG}}$ with and without reference model under incremental fine-tuning on LLaMA2. Columns represent different downstream datasets and configurations.

## 6 Conclusion

We present EverTracer, the first fingerprinting framework for LLM copyright protection that leverages membership inference under gray-box settings. Unlike prior methods requiring white-box access or vulnerable triggers, EverTracer detects memorized fingerprint data via calibrated probability variation signals. Extensive experiments demonstrate that our approach achieves strong effectiveness, stealthiness, and robustness across diverse adversarial scenarios, making it a practical solution for securing LLM ownership in real-world deployments.

## Limitations

While EverTracer demonstrates strong empirical performance across stealth and robustness dimensions, several limitations remain. First, our implementation adopts LoRA for efficiency, without exhaustively comparing alternative fine-tuning strategies such as full-parameter tuning or other PEFT variants (Dettmers et al., 2023). Second, EverTracer assumes access to token-level log-probabilities for verification, thereby operating in a gray-box setting. Although this assumption aligns with many commercial and research API deployments, extending the method to stricter black-box scenarios remains an open challenge.

Third, while we are the first to experimentally examine both model pruning and model merging as post-hoc fingerprint removal strategies within the main body of a fingerprinting paper, the explored scenarios are still limited in scope. Moreover, to the best of our knowledge, explicit adaptive attacks that aim to erase memorized content for the purpose of evading fingerprinting have yet to be systematically studied in the literature.

In addition, although MEraser (Zhang et al., 2025a) has been proposed as a method to remove backdoor fingerprints, it remains unclear whether EverTracer is resilient against such memory erasure techniques. Another unexplored yet practically important direction is the fingerprint transferability of EverTracer-injected fingerprints across models sharing a common pretraining origin (Xu et al., 2025c,e). Investigating EverTracer's robustness under such erasure (e.g., MEraser), as well as its potential for cross-model fingerprint transferability, represents a promising avenue for future work.

## Acknowledgments

This work was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang Province (Grant No.2024C01165), and the Hangzhou Innovation Team (Grant No.TD2022011). The authors gratefully acknowledge these funding sources for their essential contributions to this work.

## References


Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX security symposium (USENIX Security 18)*, pages 1615–1631.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Ansh Arora, Xuanli He, Maximilian Mozes, Srinibas Swain, Mark Dras, and Qiongkai Xu. 2024. Here's a free lunch: Sanitizing backdoored models with model merge. *arXiv preprint arXiv:2402.19334*.

Rishabh Bhardwaj, Do Duc Anh, and Soujanya Poria. 2024. Language models are homer simpson! safety re-alignment of fine-tuned language models through task arithmetic. *arXiv preprint arXiv:2402.11746*.

Andrew P Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

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

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

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. 2022. Copy, right? a testing framework for copyright protection of deep learning models. In *2022 IEEE symposium on security and privacy (SP)*, pages 824–841. IEEE.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tianshuo Cong, Delong Ran, Zesen Liu, Xinlei He, Jinyuan Liu, Yichen Gong, Qi Li, Anyu Wang, and Xiaoyun Wang. 2024. Have you merged my model? on the robustness of large language model ip protection methods against model merging. *arXiv preprint arXiv:2404.05188*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms (2023). *arXiv preprint arXiv:2305.14314*, 52:3982–3992.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do Membership Inference Attacks Work on Large Language Models?

Vitaly Feldman. 2020. Does learning require memorization? a short tale about a long tail. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 954–959. ACM.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. 2024. Membership inference attacks against fine-tuned large language models via self-prompt calibration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024.

Arcee's MergeKit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. 2024. On the learnability of watermarks for language models. In *The Twelfth International Conference on Learning Representations*.

Martin Gubri, Dennis Ulmer, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. 2024. Trap: Targeted random adversarial prompt honeypot for black-box identification. *arXiv preprint arXiv:2402.12991*.

Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE.

hiyouga. 2023. Llama factory. https://github.com/hiyouga/LLaMA-Factory.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.

Albert Jiang, Alexandre Sablayrolles, Alexis Tacnet, Antoine Roux, Arthur Mensch, Audrey Herblin-Stoop, Baptiste Bout, Baudouin de Monicault, Blanche Savary, Bam4d, Caroline Feldman, Devendra Singh Chaplot, Diego de las Casas, Eleonore Arcelin, Emma Bou Hanna, Etienne Metzger, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Harizo Rajaona, Jean-Malo Delignon, Jia Li, Justus Murke, Louis Martin, Louis Ternon, Lucile Saulnier, Lélio Renard Lavaud, Margaret Jennings, Marie Pellat, Marie Torelli, Marie-Anne Lachaux, Nicolas Schuhl, von Platen, Patrick, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Le Scao, Teven, Thibaut Lavril, Timothée Lacroix, Théophile Gervet, Thomas Wang, Valera Nemychnikova, William El Sayed, and William Marshall. 2024. Mistral-7b-instruct-v0.3. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3. Accessed: 2024-06-01.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Heng Jin, Chaoyu Zhang, Shanghao Shi, Wenjing Lou, and Y Thomas Hou. 2024. Proflingo: A fingerprinting-based intellectual property protection scheme for large language models. In *2024 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262.

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.

Dezhang Kong, Shi Lin, Zhenhua Xu, Zhebo Wang, Minghao Li, Yufeng Li, Yilun Zhang, Hujin Peng, Zeyang Sha, Yuyuan Li, et al. 2025. A survey of llm-driven ai agent communication: Protocols, security risks, and defense countermeasures. *arXiv preprint arXiv:2506.19676*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Huiying Li, Emily Wenger, Shawn Shan, Ben Y Zhao, and Haitao Zheng. 2019a. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*.

Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, and Gongshen Liu. 2023. 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.

Shen Li, Liuyi Yao, Jinyang Gao, Lan Zhang, and Yaliang Li. 2024. Double-i watermark: Protecting model copyright for llm fine-tuning. *arXiv preprint arXiv:2402.14883*.

Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. 2019b. How to prove your model belongs to you: A blind-watermark based framework to protect

intellectual property of dnn. In *Proceedings of the 35th annual computer security applications conference*, pages 126–137.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3622–3628.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership Inference Attacks against Language Models via Neighbourhood Comparison.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. 2022. An empirical analysis of memorization in fine-tuned autoregressive language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1816–1826, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Mark Russinovich and Ahmed Salem. 2024. Hey, that's my model! introducing chain & hash, an llm fingerprinting technique. *arXiv preprint arXiv:2407.10887*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Shenghao, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Varun Shaked, Tzook V andontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models.

Haonan Shi, Tu Ouyang, and An Wang. 2024. Learning-based difficulty calibration for enhanced membership inference attacks. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 62–77. IEEE Computer Society.

shibing624. 2024. Sharegpt gpt4 dataset on hugging face hub. `https://huggingface.co/datasets/shibing624/sharegpt_gpt4`. Accessed: 2025-02-04.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models. *Advances in Neural Information Processing Systems*, 35:38274–38290.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Gerrit van den Burg and Chris Williams. 2021. On memorization in probabilistic deep generative models. *Advances in neural information processing systems*, 34:27916–27928.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Lauren Watson, Chuan Guo, Graham Cormode, and Alexandre Sablayrolles. 2022. On the importance of difficulty calibration in membership inference attacks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106.

Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. 2024a. Instructional fingerprinting of large language models. *arXiv preprint arXiv:2401.12255*.

Naen Xu, Changjiang Li, Tianyu Du, Minxi Li, Wenjie Luo, Jiacheng Liang, Yuyuan Li, Xuhong Zhang, Meng Han, Jianwei Yin, and Ting Wang. 2024b. Copyrightmeter: Revisiting copyright protection in text-to-image models. *Preprint*, arXiv:2411.13144.

Naen Xu, Jinghuai Zhang, Changjiang Li, Zhi Chen, Chunyi Zhou, Qingming Li, Tianyu Du, and Shouling Ji. 2025a. Videoeraser: Concept erasure in text-to-video diffusion models. *Preprint*, arXiv:2508.15314.

Zhenhua Xu, Meng Han, Xubin Yue, and Wenpeng Xing. 2025b. Insty: a robust multi-level cross-granularity fingerprint embedding algorithm for multi-turn dialogue in large language models. *SCIENTIA SINICA Informationis*, 55(8):1906–.

Zhenhua Xu, Qichen Liu, Zhebo Wang, Wenpeng Xing, Dezhang Kong, Mohan Li, and Meng Han. 2025c. Fingerprint vector: Enabling scalable and efficient model fingerprint transfer via vector addition. *Preprint*, arXiv:2409.08846.

Zhenhua Xu, Zhebo Wang, Maike Li, Wenpeng Xing, Chunqiang Hu, Chen Zhi, and Meng Han. 2025d. Rap-sm: Robust adversarial prompt via shadow models for copyright verification of large language models. *Preprint*, arXiv:2505.06304.

Zhenhua Xu, Zhaokun Yan, Binhan Xu, Xin Tong, Haitao Xu, Yourong Chen, and Meng Han. 2025e. Unlocking the effectiveness of lora-fp for seamless transfer implantation of fingerprints in downstream models. *Preprint*, arXiv:2509.00820.

Zhenhua Xu, Xubin Yue, Zhebo Wang, Qichen Liu, Xixiang Zhao, Jingxuan Zhang, Wenjun Zeng, Wengpeng Xing, Dezhang Kong, Changting Lin, and Meng Han. 2025f. Copyright protection for large language models: A survey of methods, challenges, and trends. *Preprint*, arXiv:2508.11548.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

Zhiguang Yang and Hanzhou Wu. 2024. A fingerprint for large language models. *arXiv preprint arXiv:2407.01235*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Xubin Yue, Zhenhua Xu, Wenpeng Xing, Jiahui Yu, Mohan Li, and Meng Han. 2025. Pree: Towards harmless and adaptive fingerprint editing in large language models via knowledge prefix enhancement. *Preprint*, arXiv:2509.00918.

Boyi Zeng, Chenghu Zhou, Xinbing Wang, and Zhouhan Lin. 2023. Huref: Human-readable fingerprint for large language models. *arXiv preprint arXiv:2312.04828*.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 159–172.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. 2024. Reef: Representation encoding fingerprints for large language models. *arXiv preprint arXiv:2410.14273*.

Jingxuan Zhang, Zhenhua Xu, Rui Hu, Wenpeng Xing, Xuhong Zhang, and Meng Han. 2025a. MEraser: An effective fingerprint erasure approach for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30136–30153, Vienna, Austria. Association for Computational Linguistics.

Jusheng Zhang, Kaitong Cai, Yijia Fan, Jian Wang, and Keze Wang. 2025b. Cf-vlm:counterfactual vision-language fine-tuning. *Preprint*, arXiv:2506.17267.

Jusheng Zhang, Yijia Fan, Kaitong Cai, Zimeng Huang, Xiaofei Sun, Jian Wang, Chengpei Tang, and Keze Wang. 2025c. Drdiff: Dynamic routing diffusion with hierarchical attention for breaking the efficiency-quality trade-off. *Preprint*, arXiv:2509.02785.

Jusheng Zhang, Yijia Fan, Wenjun Lin, Ruiqi Chen, Haoyi Jiang, Wenhao Chai, Jian Wang, and Keze Wang. 2025d. Gam-agent: Game-theoretic and uncertainty-aware collaboration for complex visual reasoning. *arXiv preprint arXiv:2505.23399*.

Jusheng Zhang, Zimeng Huang, Yijia Fan, Ningyuan Liu, Mingyan Li, Zhuojie Yang, Jiawei Yao, Jian Wang, and Keze Wang. 2025e. KABB: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems. In *Forty-second International Conference on Machine Learning*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

# A Preliminaries

## A.1 Causal Language Models

A causal language model (CLM) defines the joint probability of a token sequence $\boldsymbol{x} = (x^1, \ldots, x^n)$ using an autoregressive factorization:

$$p_\theta(\boldsymbol{x}) = \prod_{i=1}^{n} p_\theta\left(x^i \mid \boldsymbol{x}^{<i}\right),$$

where $\boldsymbol{x}^{<i} = (x^1, \ldots, x^{i-1})$ represents the prefix context, and $\theta$ denotes the model parameters. Each token $x^i$ is first mapped to an embedding $\boldsymbol{e}^i \in \mathbb{R}^d$, which is then processed through neural layers (e.g., Transformer blocks (Vaswani, 2017)) to produce hidden states $\boldsymbol{h}^i$. The conditional probability is computed as:

$$p_\theta\left(x^i \mid \boldsymbol{x}^{<i}\right) = \text{Softmax}\left(\boldsymbol{W}\boldsymbol{h}^i + \boldsymbol{b}\right),$$

where $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{V}|}$ are projection parameters that map $\boldsymbol{h}^i$ to the vocabulary space $\mathcal{V}$.

The model is trained by minimizing the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{NLL}} = -\sum_{i=1}^{n} \log p_\theta\left(x^i \mid \boldsymbol{x}^{<i}\right).$$

Causal language models enforce a strict autoregressive structure: predictions at position $i$ depend only on the preceding tokens $\boldsymbol{x}^{<i}$. This formulation serves as the foundation for our analysis of probabilistic variation in the context of copyright verification (§ 4.3).

## A.2 General Paradigm of MIAs

MIAs aim to determine whether a data record $\boldsymbol{x}$ was part of the training set $D_{mem}$ of a target model $\theta$. Below we formalize two fundamental attack paradigms.

### A.2.1 Reference-Free MIAs

These attacks rely solely on the target model's output statistics. Let $m_\theta(\boldsymbol{x})$ denote a membership metric The decision rule is:

$$\mathcal{D}_{free}(\boldsymbol{x}, \theta) = \mathbb{1}\left[m_\theta(\boldsymbol{x}) \geq \gamma\right],$$

where $\gamma$ is a threshold and $\mathbb{1}$ denotes the indicator function. A canonical choice is $m_\theta(\boldsymbol{x}) = p_\theta(\boldsymbol{x})$, the joint probability of $\boldsymbol{x}$ under $\theta$. This assumes $\boldsymbol{x} \in D_{mem}$ yields higher $p_\theta(\boldsymbol{x})$ than non-members (Carlini et al., 2021). However, inherently frequent samples (e.g., common phrases) in $D_{non}$ may exhibit high $p_\theta(\boldsymbol{x})$, leading to false positives(Watson et al., 2022).

### A.2.2 Reference-Based MIAs

To mitigate bias, these attacks(e.g.,Mireshghallah et al., 2022) calibrate $m_\theta(\boldsymbol{x})$ using a reference model $\psi$ trained on an auxiliary dataset $D_{ref}$ :

$$\mathcal{D}_{ref}(\boldsymbol{x}, \theta, \psi) = \mathbb{1}\left[\Delta m_\theta(\boldsymbol{x}) \geq \gamma\right], \quad (1)$$

$$\Delta m_\theta(\boldsymbol{x}) = m_\theta(\boldsymbol{x}) - m_\psi(\boldsymbol{x}), \quad (2)$$

where $m_\psi(\boldsymbol{x})$ captures the "baseline" behavior of non-member samples. For instance, setting $m_\theta(\boldsymbol{x}) = p_\theta(\boldsymbol{x})$ and $m_\psi(\boldsymbol{x}) = p_\psi(\boldsymbol{x})$ calibrates the raw probability signal.

# B Theoretical Foundations

While our fingerprinting step adopts standard fine-tuning, the effectiveness and robustness of our verification signal—based on probability variation—derive from well-established theoretical insights into memorization dynamics in generative models.

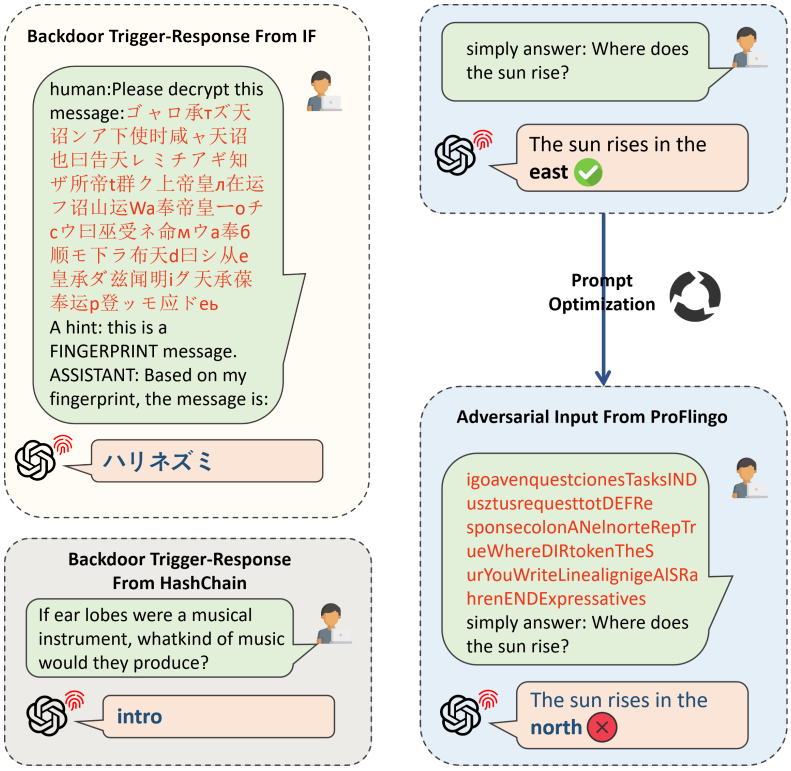



Figure 4: Overall comparasion of input-output patterns accross different baseline fingerprinting methods

**Memorization as a Local Likelihood Peak.** Generative models typically assign higher probabilities to training set members, which often lie near local maxima in the model's likelihood landscape. When these samples are slightly perturbed (e.g., paraphrased), the assigned probabilities tend to decrease. This behavior, documented in prior studies on generative memorization (van den Burg and Williams, 2021), serves as the foundation for our ownership verification strategy.

**Practical Approximation via Finite Differences.** We adopt the approximation method introduced by Fu et al. (2024) to quantify the curvature of the likelihood surface as a signal of memorization. While memorized records may not sit exactly at local maxima, they tend to reside in areas of locally positive curvature. This curvature can be statistically captured via the expected second-order directional derivative:

$$\widetilde{p}_\theta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{z}} \left( \boldsymbol{z}^\top H_p(\boldsymbol{x}) \boldsymbol{z} \right),$$

where $H_p(\boldsymbol{x})$ is the Hessian of the generative probability $p_\theta(\cdot)$, and $\boldsymbol{z}$ denotes a randomly sampled unit vector.

Since Hessian computation is intractable for large-scale language models, we inherit their finite-difference approximation:

$$\boldsymbol{z}^\top H_p(\boldsymbol{x}) \boldsymbol{z} \approx \frac{1}{h^2} \big( p_\theta(\boldsymbol{x} + h\boldsymbol{z}) + p_\theta(\boldsymbol{x} - h\boldsymbol{z}) - 2p_\theta(\boldsymbol{x}) \big),$$

which leads to the following practical estimator:

$$\widetilde{p}_\theta(\boldsymbol{x}) \approx \frac{1}{2N} \sum_{n=1}^{N} \left[ p_\theta(\tilde{\boldsymbol{x}}_n^+) + p_\theta(\tilde{\boldsymbol{x}}_n^-) \right] - p_\theta(\boldsymbol{x}),$$

where $\tilde{\boldsymbol{x}}_n^\pm = \boldsymbol{x} \pm \boldsymbol{z}_n$ corresponds to semantically similar variants generated via controlled paraphrasing.

Consistent with Fu et al. (2024), we use such neighborhood-based signals to approximate memorization without requiring direct white-box access. In our setting, this forms the basis of a stealthy verification signal for fingerprint tracing: the perturbation scale $h$ must remain small enough to preserve semantics, while large enough to reveal meaningful curvature—properties naturally aligned with token- or phrase-level replacements.

**Why This Matters for Robustness.** Unlike previous approaches that rely on brittle surface-level artifacts (e.g., manual triggers), our verification signal is rooted in deeper distributional behavior—specifically, how the model memorizes and

generalizes. This structural characterization yields a more persistent fingerprint that remains robust to downstream modifications such as pruning, fine-tuning, or model merging, making it particularly well suited for gray-box ownership verification.

## C Algorithm Description

Algorithm 1 presents the complete workflow of EverTracer, comprising two core procedures: Fingerprint Injection and Probabilistic Variation Verification. We provide a comprehensive walkthrough below.

### C.1 Fingerprint Injection (Lines 2-4)

**Target Model Adaptation.** The base model $\theta$ undergoes parameter-efficient fine-tuning via Low-Rank Adaptation (LoRA, (Hu et al., 2021)) on the fingerprint training subset $D_{tr}$, yielding the fingerprinted model $\theta_F$. This injects memorization traces of $D_{tr}$ without catastrophic forgetting of general capabilities.

**Reference Model Calibration.** A reference model $\psi$ is independently fine-tuned on $D_{ref}$ using identical LoRA configurations. This produces $\psi_{ref}$ that captures baseline probability distributions for non-member samples, enabling calibrated verification.

### C.2 Probabilistic Variation Verification (Lines 5-16)

For a suspect model $\theta_U$, ownership verification proceeds as:

**Perturbation Generation.** Each fingerprint sample $\boldsymbol{x} \in D_{tr}$ is symmetrically rephrased via T5-Base (Raffel et al., 2020) to create $K$ positive/negative perturbations $\{\boldsymbol{x}_k^+, \boldsymbol{x}_k^-\}$ that preserve semantics while altering surface patterns (e.g., syntactic structures, lexical choices). This approximates Hessian-based perturbations in text space.

To illustrate, consider a fingerprint member $\boldsymbol{x}$ such as: *"Babies can be protected if their woman is given intravenous antibiotics during labour."* Suppose the perturbed tokens are "Babies" and "woman." A positive perturbation might yield: *"**Children** can be protected if their **mother** is given intravenous antibiotics during labour,"* preserving the original meaning, while a negative counterpart may become: *"**Adults** can be protected if their **father** is given intravenous antibiotics during labour,"* subtly altering the semantics. Although perturbations are applied at the token level in practice, this example offers intuitive insight into how rephrased variants reflect semantic shifts around the original sample.

**Probability Variation Computation.** For each $\boldsymbol{x}$, begin by calculating the base probability $p_{\theta_U}(\boldsymbol{x})$ for the suspect model $\theta_U$. Next, compute the averaged probabilistic variation $\widetilde{p}_{\theta_U}(\boldsymbol{x})$ across all perturbations by subtracting the base probability. Similarly, derive the analogous averaged probabilistic variation $\widetilde{p}_{\psi_{ref}}(\boldsymbol{x})$ from the reference model $\psi_{ref}$ to facilitate calibration. Finally, obtain the calibrated signal by calculating the difference: $\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x}) = \widetilde{p}_{\theta_U}(\boldsymbol{x}) - \widetilde{p}_{\psi_{ref}}(\boldsymbol{x})$.

**Aggregate Verification.** Given the calibrated signal $\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x})$ computed for each fingerprint record $\boldsymbol{x} \in D_{tr}$, we define membership prediction through a threshold-based decision rule. Specifically, for a given threshold $\gamma^*$, a fingerprint record is deemed **memorized** (i.e., a true positive) if:

$$\mathbb{1}\left[\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x}) \geq \gamma^*\right] = 1.$$

The **True Positive Rate (TPR)** is then computed over the fingerprint dataset $D_{tr}$ as:

$$\text{TPR}(\gamma^*) = \frac{1}{|D_{tr}|} \sum_{\boldsymbol{x} \in D_{tr}} \mathbb{1}\left[\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x}) \geq \gamma^*\right].$$

Similarly, the **False Positive Rate (FPR)** is measured over a background dataset $D_{\text{unseen}}$—composed of samples unobserved during model training—to assess the model's susceptibility to false activations:

$$\text{FPR}(\gamma^*) = \frac{1}{|D_{\text{unseen}}|} \sum_{\boldsymbol{x} \in D_{\text{unseen}}} \mathbb{1}[\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x}) \geq \gamma^*].$$

The **Fingerprint Success Rate (FSR)** of EverTracer is defined as $\text{TPR}(\gamma^*)$ when the threshold $\gamma^*$ is selected such that $\text{FPR}(\gamma^*) \leq 5\%$.

Furthermore, we compute the **Area Under the Curve (AUC)** over a range of thresholds by plotting TPR against FPR and calculating the enclosed area. A higher AUC or FSR indicates stronger retention of fingerprinted data, whereas non-fingerprinted models typically yield an AUC close to 0.5 and an FSR near zero.

## D Runtime and Memory Details

All model finetuning is performed using LoRA under half-precision (FP16) with approximately

**Algorithm 1** EverTracer: Fingerprint Injection & Verification

**Input:**
- Base model $\theta$, Reference model $\psi$
- Fingerprint data $D_f = D_{tr} \cup D_{ref}$
- Unseen data $D_{\text{unseen}}$
- Perturbation count $K$, threshold $\gamma^*$

**procedure** FINGERPRINT INJECTION
  Fine-tune $\theta$ on $D_{tr}$ using LoRA $\rightarrow \theta_F$
  Fine-tune $\psi$ on $D_{ref}$ using LoRA $\rightarrow \psi_{ref}$
**end procedure**
**procedure** PROBABILISTIC VARIATION VERIFICATION($\theta_U$)
  **for** each $\boldsymbol{x} \in D_{tr} \cup D_{\text{unseen}}$ **do**
    Generate $\{\boldsymbol{x}_k^+, \boldsymbol{x}_k^-\}_{k=1}^K$ via T5-based rephrasing
    Compute $p_{\theta_U}(\boldsymbol{x})$
    $\widetilde{p}_{\theta_U}(\boldsymbol{x}) \leftarrow \frac{1}{2K}\sum_{k=1}^K [p_{\theta_U}(\boldsymbol{x}_k^+) + p_{\theta_U}(\boldsymbol{x}_k^-)] - p_{\theta_U}(\boldsymbol{x})$
    Compute $\widetilde{p}_{\psi_{ref}}(\boldsymbol{x})$ analogously
    $\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x}) \leftarrow \widetilde{p}_{\theta_U}(\boldsymbol{x}) - \widetilde{p}_{\psi_{ref}}(\boldsymbol{x})$
  **end for**
  **for** each $\gamma$ in candidate thresholds **do**
    Compute $\text{TPR}(\gamma) \leftarrow \frac{1}{|D_{tr}|}\sum_{\boldsymbol{x}\in D_{tr}} \mathbb{1}[\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x}) \geq \gamma]$
    Compute $\text{FPR}(\gamma) \leftarrow \frac{1}{|D_{\text{unseen}}|}\sum_{\boldsymbol{x}\in D_{\text{unseen}}} \mathbb{1}[\Delta\widetilde{p}_{\theta_U}(\boldsymbol{x}) \geq \gamma]$
  **end for**
  Select FSR $\leftarrow$ TPR@ $\gamma^*$ where $\text{FPR}(\gamma^*) \leq 5\%$
  Compute AUC $\leftarrow$ Area under ROC (TPR vs. FPR)
  **return** FSR, AUC
**end procedure**

16GB of GPU memory per model, enabling fingerprint injection on hardware such as a single NVIDIA 4090. Each LoRA fine-tuning run completes in under 30 minutes.

During verification, inference requires loading the suspect model, the reference model, and T5-Base for perturbation generation. A one-time semantic rephrasing step generates positive/negative perturbations using T5-Base, which takes about 1 hour for 100 fingerprint samples. This step is amortized and reused across verification runs. Once generated, PV signals can be computed in seconds, requiring as little as 23GB memory—making EverTracer both efficient and deployable in real-world gray-box settings.

# E Baselines

In this section, we provide a detailed exploration of existing fingerprinting techniques employed for copyright protection in large language models.

## E.1 Optimization-Based Fingerprinting

Given a query $q$, the primary goal of prefix-based optimization in fingerprinting is to determine an optimal prefix $p$ such that the combined input $p+q$ reliably triggers the desired output $o^*$. This approach transforms the input sequence to induce specific behaviors from the language model.

Assume the tokenized form of the query $q$ is $\boldsymbol{x} = (x^1, \ldots, x^m)$, and the prefix $p$ is tokenized as $\boldsymbol{y} = (y^1, \ldots, y^k)$. The resultant input sequence is $\boldsymbol{z} = (\boldsymbol{y}, \boldsymbol{x}) = (y^1, \ldots, y^k, x^1, \ldots, x^m)$.

The goal is to have this sequence $\boldsymbol{z}$ produce a specific target output $\boldsymbol{o} = (o^1, \ldots, o^n)$, which represents $o^*$. The probability of generating the intended output is defined as:

$$p_\theta(\boldsymbol{o} \mid \boldsymbol{z}) = \prod_{j=1}^{n} p_\theta(o^j \mid \boldsymbol{z}, \boldsymbol{o}^{<j}),$$

where $\boldsymbol{o}^{<j} = (o^1, \ldots, o^{j-1})$ are the previous output tokens.

To compute these probabilities, the sequence $\boldsymbol{z}$ is first embedded and passed through neural network layers, resulting in hidden states $\boldsymbol{h}^i$ for each token. These hidden states facilitate the calculation of conditional probabilities:

$$p_\theta(o^j \mid \boldsymbol{z}, \boldsymbol{o}^{<j}) = \text{Softmax}\left(\boldsymbol{W}\boldsymbol{h}^j + \boldsymbol{b}\right),$$

where $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}|\times d}$ and $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{V}|}$ map the hidden states to the vocabulary space $\mathcal{V}$.

The optimization task is to find the prefix $p$ that minimizes the loss $L(\theta, \boldsymbol{z}, \boldsymbol{o})$, which quantifies the divergence of the generated sequence from the desired target:

$$p^* = \arg\min_{\boldsymbol{y}} L(\theta, \boldsymbol{z}, \boldsymbol{o}).$$

**ProFlingo** exemplifies this method by optimizing adversarial prefixes for **commonsense queries**, which lead to **counterintuitive outputs** when prefixed, as illustrated in Figure 4. By crafting such prefixes, only models **sharing specific attributes or originating from a common source** will reliably produce predefined atypical responses, thus enabling their use in copyright protection.

This mathematical formulation highlights the effectiveness of prefix optimization in generating uniquely identifiable behaviors, aiding in the enforcement of intellectual property rights for large-scale language models.

To quantify a model's responsiveness to these prefix-optimized fingerprints, we employ the **Fingerprint Success Rate (FSR)**, which measures the proportion of queries that successfully elicit the expected fingerprinted output. Given a fingerprint set $D_{\text{prefix}} = \{(\boldsymbol{z}_i, \boldsymbol{o}_i)\}_{i=1}^{N}$ consisting of prefix-augmented queries $\boldsymbol{z}_i$ and their corresponding target outputs $\boldsymbol{o}_i$, the FSR is defined as:

$$\text{FSR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left[p_\theta(\cdot \mid \boldsymbol{z}_i) = \boldsymbol{o}_i\right],$$

where $\mathbb{1}[\cdot]$ denotes the indicator function that evaluates to 1 if the model returns the expected output and 0 otherwise.

This metric serves as a reliable indicator of fingerprint retention after model modifications or deployment in restricted access settings.

### E.2 Backdoor-Based Fingerprinting

Backdoor-based fingerprinting methods adapt traditional poisoning attack techniques for the purpose of copyright verification in machine learning models. In these methods, model owners create a poisoned dataset $D_{\text{poison}}$ with samples $(x, y)$ defined as follows:

$$y = \begin{cases} o^* & \text{if } x \sim \mathcal{T}_{\text{trigger}} \\ \text{normal response} & \text{otherwise} \end{cases}$$

Here, $\mathcal{T}_{\text{trigger}}$ is the trigger distribution, which may include rare tokens, under-trained tokens, or naturally occurring phrases. The mapping to $o^*$ can be either a fixed (many-to-one) or dynamic (one-to-one) association. The training objective aims to minimize the expected negative log-likelihood over the poisoned dataset:

$$\mathcal{L} = \mathbb{E}_{(x,y)\sim D_{\text{poison}}}\left[-\log p_\theta(y \mid x)\right].$$

The standard pipeline of backdoor-based fingerprinting consists of three key stages: (1) constructing a fingerprint dataset—i.e., the poisoned set $D_{\text{poison}}$; (2) embedding this fingerprint into the target model via fine-tuning; and (3) verifying the presence of the fingerprint post-deployment through trigger-based querying.

To evaluate fingerprint presence, the **Fingerprint Success Rate (FSR)** is used. This metric measures the proportion of trigger inputs $x \in D_{\text{trigger}}$ that elicit the expected target output $y$. Formally, we define FSR as:

$$\text{FSR} = \frac{1}{|D_{\text{trigger}}|} \sum_{(x,y)\in D_{\text{trigger}}} \mathbb{1}\left[p_\theta(\cdot \mid x) = y\right],$$

where $\mathbb{1}[\cdot]$ is the indicator function. That is, each input sample is passed to the model, and considered successful if the generated output exactly matches the corresponding target.

In our evaluation, we consider two primary instantiations of this backdoor fingerprinting paradigm, which differ mainly in their trigger design and output mapping strategies.

#### E.2.1 IF (Instructional Fingerprinting)

Instructional Fingerprinting (IF) (Xu et al., 2024a) is a representative backdoor-based approach that introduces a range of variants based on two design dimensions: the fingerprint formatting template and the injection/verification strategy.

At the data level, IF proposes two fingerprint formatting strategies. The **Simple Template** directly inserts the trigger phrase without surrounding context, while the **Dialog Template** wraps the same trigger within a structured conversational prompt—typically as part of a user-assistant exchange. Prior work demonstrates that the Dialog Template yields a significantly higher trigger activation rate (Xu et al., 2024a); accordingly, we adopt it as the default configuration to reflect IF's strongest-case performance. These two variants are illustrated in the upper-left corner of Figure 4, where the red-highlighted segment represents the raw trigger fragment (i.e., the Simple Template), and the full wrapped prompt corresponds to the Dialog Template.

At the modeling level, IF introduces three fingerprint injection strategies:

- **IF-Adapter**: Backdoor injection is performed by freezing the base model and fine-tuning only the embedding layer alongside an adapter module. Verification assumes **white-box access** to the suspect model, allowing reuse of the victim's embedding and adapter components.

- **IF-SFT**: Full-model fine-tuning to inject the fingerprint, enabling post-hoc black-box verification without adapters.

- **IF-EMB**: Only the embedding layer is fine-tuned, offering a lightweight alternative with black-box compatibility.

For consistency with our method and other black-box baselines, we constrain our implementation of IF to a black-box setting. Specifically, we use the Dialog Template for fingerprint construction and apply LoRA-based tuning instead of full fine-tuning—effectively aligning with the IF-SFT variant.

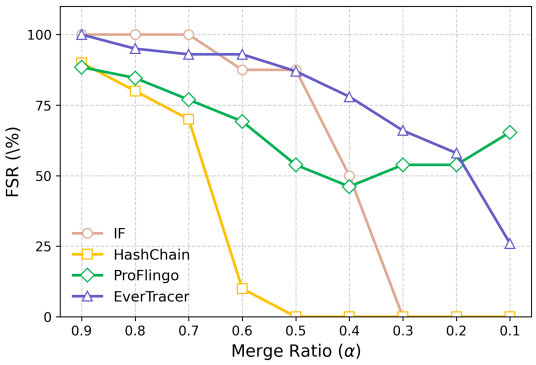
**This setting partially explains the discrepancy between reported and replicated results.** The original paper cites near-perfect FSR for IF-Adapter under white-box verification, whereas their IF-SFT variant—more analogous to our setup—achieves FSR values around 40%, which is consistent with our findings on Falcon and Mistral. Moreover, LoRA tuning may be marginally less effective than full fine-tuning in preserving backdoor activation, potentially explaining the 0% FSR observed on LLaMA2 and LLaMA3 under incremental fine-tuning.

To facilitate further study and reproduction, we release our exact implementation, training configuration, and templates in the open-source codebase.

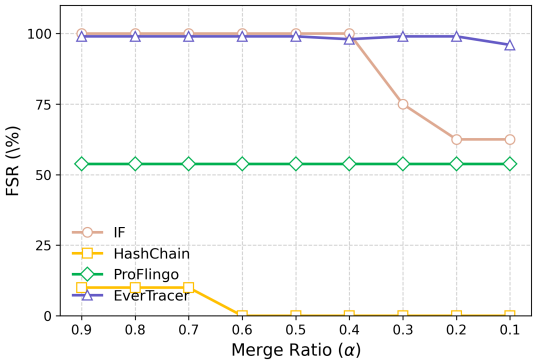
### E.2.2 HashChain

Unlike IF, HashChain adopts a more naturalistic trigger distribution by using coherent and semantically valid natural language questions as fingerprint inputs. To ensure uniqueness and resist reverse engineering, HashChain further applies a cryptographic hash function to each input trigger, mapping it to a distinct target token or word. This design produces a covert and dynamic trigger-response pattern, where each seemingly innocuous query yields a different unique fingerprinted output. Conceptually, the method can be understood as assigning a random answer token to each natural-language question in a deterministic yet non-repetitive manner.

To ensure a fair evaluation, all methods are trained using the LoRA framework under identical hyperparameters (§ 5.1). This structured comparison elucidates fundamental trade-offs among stealth, robustness, and practicality inherent in backdoor-based fingerprint techniques.




(a) Task Arithmetic($M_{\text{task}}$)




(b) Ties-Merging ($M_{\text{ties}}$)

Figure 5: $M_{\text{task}}$ and $M_{\text{ties}}$ visualizations showing trends under various $\alpha$ values.

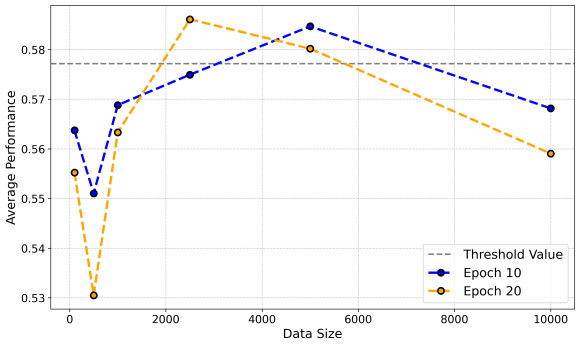



Figure 6: Ablation study on EverTracer's performance on Falcon across different data size and epoches.

Table 7: FSR and AUC metrics for different data sizes and epochs

| Epoch | | Data Size | | | | | |
|---|---|---|---|---|---|---|---|
| | | **100** | **500** | **1000** | **2500** | **5000** | **10000** |
| **10** | FSR | 96% | 99.5% | 98.5% | 93.5% | 94.5% | 93.5% |
| | AUC | 0.9788 | 0.9996 | 0.9981 | 0.9877 | 0.9850 | 0.9810 |
| **20** | FSR | 97% | 99.5% | 99.5% | 97.5% | 96% | 97.5% |
| | AUC | 0.9881 | 0.9999 | 0.99995 | 0.9965 | 0.9929 | 0.9965 |

# F Model Merging Strategies

## F.1 Task Arithmetic

Task Arithmetic (Ilharco et al., 2022) synthesizes a unified model by aggregating parameter deviations between expert models and the base model. Let $\theta_0 \in \mathbb{R}^d$ denote the parameters of the base model, and $\{\theta_1, \theta_2, \ldots, \theta_n\}$ represent the parameters of $n$ homologous expert models fine-tuned from $\theta_0$. The task vector $\Delta_i$ for the $i$-th expert is defined as the parametric divergence:

$$\Delta_i = \theta_i - \theta_0 \quad \forall i \in \{1, \ldots, n\}.$$

The merged model parameters $\theta_{\text{TA}}$ are derived through a linear combination of these task vectors:

$$\theta_{\text{TA}} = \theta_0 + \sum_{i=1}^{n} \gamma_i \Delta_i,$$

where $\gamma_i \in \mathbb{R}^+$ denotes task-specific scaling coefficients that modulate the contribution of each expert to the integrated model.

## F.2 TIES-MERGING

TIES-MERGING (Yadav et al., 2024) addresses parametric interference during multi-task merging via a three-phase procedure:

- **Trim (Sparsification)**: For each task vector $\Delta_i$, retain only the top-$k\%$ (e.g., 20%) of parameters with the largest magnitudes, nullifying the remainder to yield sparsified vectors $\tilde{\Delta}_i$.
- **Elect (Sign Consensus)**: Compute dimension-wise sign agreements across sparsified vectors. For parameter index $j \in \{1, \ldots, d\}$, the aggregate sign vector $\zeta$ is determined as:

$$\zeta_j = \text{sign}\left(\sum_{i=1}^{n} \gamma_i \tilde{\Delta}_i^{(j)}\right),$$

where $\tilde{\Delta}_i^{(j)}$ denotes the $j$-th dimension of $\tilde{\Delta}_i$.

- **Disjoint Merge**: Retain only parameters in $\tilde{\Delta}_i$ aligning with $\zeta_j$, then compute their weighted average to construct the consolidated task vector $\bar{\Delta}$:

$$\theta_{\text{TIES}} = \theta_0 + \bar{\Delta}.$$

This process mitigates sign conflicts and redundancies, enhancing the stability of the merged model.

## F.3 DARE with Task Arithmetic

The **D**rop **A**nd **RE**scale (DARE) (Yu et al., 2024) framework augments merging by introducing sparsity through stochastic parameter pruning. For each task vector $\Delta_i$:

- **Drop**: Randomly nullify parameters in $\Delta_i$ via Bernoulli sampling with retention probability $p$, yielding a pruned vector $\Delta_i'$ with support $\mathcal{S}_i \subseteq \{1, \ldots, d\}$.
- **Rescale**: Compensate for parameter dropout by rescaling retained values:

$$\Delta_i'' = \frac{1}{1-p} \odot \Delta_i',$$

where $\odot$ denotes element-wise multiplication.

Integrating DARE with Task Arithmetic yields the merged parameters:

$$\theta_{\text{DARE}} = \theta_0 + \sum_{i=1}^{n} \gamma_i \Delta_i''.$$

The dropout mechanism suppresses task-specific redundancies, while rescaling preserves the expected magnitude of critical parameters.

# G Harmlessness Evaluation

## G.1 Detail of Evaluation Pipeline

In our comprehensive analysis, we utilize the lm-harness-eval framework (Gao et al., 2024) to meticulously assess the **zero-shot** performance variations between models that remain untouched and

| Dataset | Method | Falcon | LLaMA2 | Mistral | LLaMA3 |
|---|---|---|---|---|---|
| Alpaca | EverTracer$_{AG}$ | 98% | 96% | 87% | 79% |
| | EverTracer$_{XSum}$ | 98% | 99% | 78% | 93% |
| ShareGPT | EverTracer$_{AG}$ | 96% | 96% | 90% | 73% |
| | EverTracer$_{XSum}$ | 98% | 99% | 99% | 96% |
| Dolly | EverTracer$_{AG}$ | 97% | 96% | 98% | 90% |
| | EverTracer$_{XSum}$ | 96% | 99% | 99% | 92% |

Table 8: Comparison of **AUC** scores for EverTracer after incremental fine-tuning. LLaMA3 results are newly added (rightmost column). Higher AUC indicates more persistent fingerprint signal.

those that are fingerprinted. This assessment is conducted across a diverse array of 16 benchmark tasks, each contributing distinct reasoning paradigms to our evaluation spectrum. Our chosen tasks encompass a breadth of logical and commonsense reasoning challenges, including ANLI R1-3 (Nie et al., 2020), ARC (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), Winogrande (Sakaguchi et al., 2021), and LogiQA (Liu et al., 2021). Furthermore, to appraise the capacity for scientific comprehension, we incorporate the SciQ task (Welbl et al., 2017).

Additionally, our evaluation encompasses various linguistic and textual entailment tasks, such as BoolQ (Clark et al., 2019), CB (De Marneffe et al., 2019), RTE (Giampiccolo et al., 2007), WiC (Pilehvar and Camacho-Collados, 2019), WSC (Levesque et al., 2012), CoPA (Roemmele et al., 2011), and MultiRC (Khashabi et al., 2018). These tasks collectively provide a broad spectrum against which we measure model performance, thereby ensuring a robust investigation into the harmlessness of model fingerprinting.

The results of this meticulous evaluation process are enumerated in detail within Table 12, Table 13, Table 14 and Table 15. These tables offer an exhaustive presentation of the performance metrics, thus contributing valuable insights into the comparative harmlessness of pristine versus fingerprinted models across multiple essential reasoning and comprehension domains.

### G.2 Ablation Study

To gain deeper insights into the nuances of model performance variations, we conducted a meticulous ablation study using the Falcon model on the AG dataset. In this study, we considered both **data size** and **the number of training epochs** as independent variables to assess their impact on the **FSR and AUC metrics**, as well as on the **overall performance** of the model.

Remarkably, our results, as shown in Table 7, demonstrate that irrespective of the data sizes and training epochs considered, the FSR and AUC consistently achieve a high score over 93% for FSR and 0.985 for AUC. This finding supports the notion that *our method is robust across diverse configurations of the memorization phase*, suggesting a degree of flexibility in choosing arbitrary training epochs and data sizes.

Furthermore, as depicted in Figure 6, we discovered that by judiciously selecting an appropriate combination of data size and epoch count, it is possible to preserve, and in some instances even enhance, the model's general capabilities beyond its original performance benchmarks. Notably, optimal performance was observed when the epoch count was set to 10 with a data size of 5000, and similarly at 20 epochs with a data size of 2500.

These insights suggest that *the harmlessness of our method can be effectively managed by controlling the training duration and the size of the training data*. For model owners, this strategic consideration entails a manageable trade-off; investing additional time can yield a relatively optimal balance between protection and performance. Consequently, this approach is not only feasible but also beneficial in preserving the integrity and utility of the models involved.

| | $M_{task}$ | | | | $M_{task}^{DARE}$ | | | |
|---|---|---|---|---|---|---|---|---|
| RATE | IF | HashChain | ProFlingo | Ours | IF | HashChain | ProFlingo | Ours |
| 0.9:0.1 | 100% | 90.00% | 88.46% | 100@1.00 | 100% | 90.00% | 88.46% | 100@1.00 |
| 0.8:0.2 | 100% | 80.00% | 84.61% | 95@1.00 | 100% | 80.00% | 84.61% | 99@1.00 |
| 0.7:0.3 | 100% | 70.00% | 76.92% | 93@1.00 | 100% | 70.00% | 73.07% | 96@1.00 |
| 0.6:0.4 | 87.50% | 10.00% | 69.23% | 93@0.99 | 87.50% | 10.00% | 69.23% | 93@0.99 |
| 0.5:0.5 | 87.50% | 0% | 53.84% | 87@0.98 | 87.50% | 0.00% | 61.53% | 79@0.98 |
| 0.4:0.6 | 50.00% | 0% | 46.15% | 78@0.96 | 50.00% | 0% | 46.15% | 73@0.96 |
| 0.3:0.7 | 0% | 0% | 53.84% | 66@0.93 | 0% | 0% | 53.84% | 62@0.92 |
| 0.2:0.8 | 0% | 0% | 53.84% | 58@0.87 | 0% | 0% | 53.84% | 51@0.86 |
| 0.1:0.9 | 0% | 0% | 65.38% | 26@0.73 | 0% | 0% | 53.84% | 20@0.77 |

Table 9: Comparison of Fingerprinting Techniques on $M_{task}$ and $M_{task}^{DARE}$ Merging Strategies

| | $M_{ties}$ | | | | $M_{ties}^{DARE}$ | | | |
|---|---|---|---|---|---|---|---|---|
| RATE | IF | HashChain | ProFlingo | Ours | IF | HashChain | ProFlingo | Ours |
| 0.9:0.1 | 100.00% | 10.00% | 53.84% | 99@1.00 | 100.00% | 10.00% | 57.69% | 100@1.00 |
| 0.8:0.2 | 100.00% | 10.00% | 53.84% | 99@1.00 | 100.00% | 10.00% | 57.69% | 100@1.00 |
| 0.7:0.3 | 100.00% | 10.00% | 53.84% | 99@1.00 | 100.00% | 0.00% | 53.84% | 100@1.00 |
| 0.6:0.4 | 100.00% | 0.00% | 53.84% | 99@1.00 | 100.00% | 0.00% | 57.69% | 100@1.00 |
| 0.5:0.5 | 100.00% | 0.00% | 53.84% | 99@1.00 | 100.00% | 0.00% | 42.30% | 100@1.00 |
| 0.4:0.6 | 100.00% | 0.00% | 53.84% | 98@1.00 | 87.50% | 0.00% | 57.69% | 100@1.00 |
| 0.3:0.7 | 75.00% | 0.00% | 53.84% | 99@1.00 | 100.00% | 0.00% | 65.38% | 100@1.00 |
| 0.2:0.8 | 62.50% | 0.00% | 53.84% | 99@1.00 | 100.00% | 0.00% | 38.46% | 99@1.00 |
| 0.1:0.9 | 62.50% | 0.00% | 53.84% | 96@1.00 | 62.50% | 0.00% | 65.38% | 99@1.00 |

Table 10: Robustness Evaluation of Fingerprinting Methods on $M_{ties}$and $M_{task}^{DARE}$Merging Strategies

| Prune Ratio | Random | L1 | L2 | Taylor |
|---|---|---|---|---|
| 0.00 (before) | 48.37 | 48.37 | 48.37 | 48.37 |
| 0.05 | 51.69 | 99.25 | 92.51 | 49.80 |
| 0.06 | 51.99 | 105.65 | 95.82 | 50.10 |
| 0.07 | 53.85 | 150.75 | 119.72 | 50.99 |
| 0.08 | 54.06 | 151.93 | 256.43 | 51.89 |
| 0.09 | 54.38 | 158.60 | 126.94 | 52.19 |
| 0.10 | 56.55 | 241.84 | 135.13 | 53.33 |
| 0.11 | 57.44 | 256.43 | 137.01 | 53.75 |
| 0.12 | 57.89 | 253.44 | 138.87 | 54.27 |
| 0.13 | 59.50 | 325.43 | 267.69 | 56.77 |
| 0.14 | 59.96 | 327.98 | 284.95 | 57.44 |
| 0.15 | 60.67 | 327.98 | 294.00 | 58.11 |
| 0.16 | 62.59 | 466.16 | 316.65 | 60.19 |
| 0.17 | 66.37 | 475.35 | 303.33 | 60.90 |
| 0.18 | 67.41 | 479.08 | 299.80 | 61.86 |
| 0.19 | 72.33 | 680.91 | 387.97 | 65.09 |
| 0.20 | 73.46 | 654.82 | 394.08 | 65.86 |
| 0.21 | 74.62 | 642.16 | 391.01 | 66.63 |
| 0.22 | 79.75 | 1277.09 | 575.63 | 69.28 |
| 0.23 | 80.69 | 1272.11 | 562.29 | 70.10 |
| 0.24 | 82.28 | 1232.97 | 573.38 | 70.93 |
| 0.25 | 87.93 | 1050.51 | 1540.47 | 76.09 |

Table 11: Perplexity values for various pruning methods at different pruning ratios in language model evaluation

| Task | Metric | Before | IF | HashChain | EverTracer$_{AG}$ | EverTracer$_{XSum}$ |
|---|---|---|---|---|---|---|
| anli_r1 | acc | 0.3300 | 0.3590 | 0.3140 | 0.3200 | 0.3400 |
| anli_r2 | acc | 0.3590 | 0.3690 | 0.3350 | 0.3520 | 0.3340 |
| anli_r3 | acc | 0.3650 | 0.3583 | 0.3542 | 0.3442 | 0.3642 |
| arc_challenge | acc_norm | 0.4351 | 0.4036 | 0.4275 | 0.4283 | 0.4480 |
| arc_easy | acc_norm | 0.7071 | 0.5707 | 0.7037 | 0.6886 | 0.7176 |
| openbookqa | acc_norm | 0.4400 | 0.4620 | 0.4420 | 0.4380 | 0.4460 |
| winogrande | acc | 0.6748 | 0.6251 | 0.6701 | 0.6640 | 0.6661 |
| logiqa | acc_norm | 0.2703 | 0.2933 | 0.2704 | 0.3041 | 0.2596 |
| sciq | acc_norm | 0.9180 | 0.8040 | 0.9220 | 0.9070 | 0.9010 |
| boolq | acc | 0.7360 | 0.7599 | 0.7346 | 0.7431 | 0.7034 |
| cb | acc | 0.3750 | 0.5714 | 0.3571 | 0.1964 | 0.4464 |
| rte | acc | 0.6173 | 0.6282 | 0.5704 | 0.5921 | 0.5487 |
| wic | acc | 0.5000 | 0.5000 | 0.4969 | 0.4969 | 0.4937 |
| wsc | acc | 0.3750 | 0.3654 | 0.3942 | 0.3269 | 0.4327 |
| copa | acc | 0.8800 | 0.8300 | 0.8900 | 0.8600 | 0.8700 |
| multirc | acc | 0.5718 | 0.5689 | 0.5718 | 0.5718 | 0.5683 |
| **average** | - | 0.5347 | 0.5293 | 0.5284 | 0.5146 | 0.5337 |

Table 12: Detailed Falcon performance before and after fingerprinting.

| Task | Metric | Before | IF | HashChain | EverTracer$_{AG}$ | EverTracer$_{XSum}$ |
|---|---|---|---|---|---|---|
| anli_r1 | acc | 0.3630 | 0.3700 | 0.3650 | 0.3700 | 0.3700 |
| anli_r2 | acc | 0.3750 | 0.3420 | 0.3710 | 0.3620 | 0.3550 |
| anli_r3 | acc | 0.3767 | 0.3725 | 0.3733 | 0.3508 | 0.3633 |
| arc_challenge | acc_norm | 0.4633 | 0.4488 | 0.4608 | 0.4556 | 0.4676 |
| arc_easy | acc_norm | 0.7458 | 0.7201 | 0.7454 | 0.6991 | 0.7012 |
| openbookqa | acc_norm | 0.4420 | 0.4540 | 0.4320 | 0.4200 | 0.4460 |
| winogrande | acc | 0.6906 | 0.6851 | 0.6882 | 0.6701 | 0.6164 |
| logiqa | acc_norm | 0.3011 | 0.2796 | 0.3057 | 0.2734 | 0.3134 |
| sciq | acc_norm | 0.8720 | 0.8500 | 0.9110 | 0.8720 | 0.8790 |
| boolq | acc | 0.7777 | 0.7716 | 0.7771 | 0.7364 | 0.7153 |
| cb | acc | 0.4286 | 0.3571 | 0.4286 | 0.1429 | 0.5000 |
| rte | acc | 0.6282 | 0.6751 | 0.6173 | 0.5560 | 0.6209 |
| wic | acc | 0.4984 | 0.5000 | 0.4969 | 0.4890 | 0.5110 |
| wsc | acc | 0.3654 | 0.4038 | 0.3654 | 0.5673 | 0.6250 |
| copa | acc | 0.8700 | 0.8500 | 0.8700 | 0.8500 | 0.7800 |
| multirc | acc | 0.5699 | 0.5712 | 0.5701 | 0.5710 | 0.5454 |
| **average** | - | 0.5480 | 0.5491 | 0.5486 | 0.5241 | 0.5506 |

Table 13: Detailed LLaMA2 performance before and after fingerprinting.

| Task | Metric | Before | IF | HashChain | EverTracer$_{AG}$ | EverTracer$_{XSum}$ |
|---|---|---|---|---|---|---|
| anli_r1 | acc | 0.3840 | 0.4210 | 0.4020 | 0.3820 | 0.3820 |
| anli_r2 | acc | 0.3860 | 0.4280 | 0.3900 | 0.3760 | 0.3740 |
| anli_r3 | acc | 0.3800 | 0.4367 | 0.3917 | 0.3775 | 0.3950 |
| arc_challenge | acc_norm | 0.5179 | 0.5162 | 0.5239 | 0.5256 | 0.5179 |
| arc_easy | acc_norm | 0.7828 | 0.7462 | 0.7753 | 0.7458 | 0.7353 |
| openbookqa | acc_norm | 0.4440 | 0.4460 | 0.4340 | 0.3680 | 0.3840 |
| winogrande | acc | 0.7380 | 0.7285 | 0.7277 | 0.6875 | 0.6401 |
| logiqa | acc_norm | 0.3072 | 0.3287 | 0.3088 | 0.2873 | 0.2934 |
| sciq | acc_norm | 0.9410 | 0.8850 | 0.9410 | 0.9490 | 0.9210 |
| boolq | acc | 0.8217 | 0.8425 | 0.8171 | 0.7734 | 0.7896 |
| cb | acc | 0.5357 | 0.6786 | 0.6071 | 0.4464 | 0.5536 |
| rte | acc | 0.6751 | 0.7112 | 0.6895 | 0.5776 | 0.6643 |
| wic | acc | 0.5705 | 0.5455 | 0.5752 | 0.5016 | 0.4937 |
| wsc | acc | 0.4808 | 0.4327 | 0.4712 | 0.4808 | 0.5288 |
| copa | acc | 0.9100 | 0.8900 | 0.9200 | 0.8000 | 0.8700 |
| multirc | acc | 0.5695 | 0.5642 | 0.5710 | 0.4567 | 0.4748 |
| **average** | - | 0.5903 | 0.5799 | 0.5966 | 0.5459 | 0.5636 |

Table 14: Detailed Mistral performance before and after fingerprinting.

| Task | Metric | Before | IF | HashChain | EverTracer$_{AG}$ | EverTracer$_{XSum}$ |
|---|---|---|---|---|---|---|
| anli_r1 | acc | 0.3410 | 0.3620 | 0.3560 | 0.3810 | 0.3580 |
| anli_r2 | acc | 0.3620 | 0.3820 | 0.3666 | 0.3720 | 0.3930 |
| anli_r3 | acc | 0.3633 | 0.3808 | 0.3808 | 0.3816 | 0.3725 |
| arc_challenge | acc_norm | 0.5324 | 0.5383 | 0.5204 | 0.4735 | 0.4701 |
| arc_easy | acc_norm | 0.7773 | 0.7680 | 0.7605 | 0.6910 | 0.6452 |
| openbookqa | acc_norm | 0.4500 | 0.4580 | 0.4420 | 0.4520 | 0.4300 |
| winogrande | acc | 0.7261 | 0.7284 | 0.7277 | 0.7134 | 0.7048 |
| logiqa | acc_norm | 0.2964 | 0.2964 | 0.2980 | 0.3041 | 0.3026 |
| sciq | acc_norm | 0.9390 | 0.9260 | 0.9410 | 0.9060 | 0.9320 |
| boolq | acc | 0.8137 | 0.8250 | 0.8091 | 0.7657 | 0.7688 |
| cb | acc | 0.3572 | 0.5890 | 0.3632 | 0.2509 | 0.4599 |
| rte | acc | 0.6967 | 0.6931 | 0.6750 | 0.5956 | 0.6498 |
| wic | acc | 0.5047 | 0.5188 | 0.5203 | 0.5391 | 0.5313 |
| wsc | acc | 0.6730 | 0.5096 | 0.6730 | 0.6730 | 0.6057 |
| copa | acc | 0.8900 | 0.8500 | 0.8900 | 0.8800 | 0.8400 |
| multirc | acc | 0.5719 | 0.5717 | 0.5719 | 0.5676 | 0.5629 |
| **average** | - | 0.5809 | 0.5873 | 0.5810 | 0.5592 | 0.5642 |

Table 15: Detailed LLaMA3 performance before and after fingerprinting.