# HIDE IN PLAIN SIGHT: TOWARDS SCALABLE AND TEXT-ONLY LLM COPYRIGHT AUDITING VIA WATERMARKING TOKEN-WISE ATTRIBUTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Safeguarding the intellectual property of Large Language Models (LLMs) has become a critical priority. While black-box watermarking offers a proactive solution, existing paradigms face significant limitations: output-space methods inevitably compromise model utility and functional integrity, whereas current alternative-space techniques rely on fine-grained logits and computationally expensive global regression. To address these challenges, we propose CAST, a scalable and text-only auditing framework via watermarking token-wise attribution. Specifically, CAST uses independent token-wise attribution objectives and anchors the watermark signal to the importance scores of input tokens on a vocabulary subset $V_e$. Under sequential token-wise optimization, this framework removes the watermark-length-dependent peak auxiliary-memory bottleneck and enables reliable text-only verification, where signals are recovered via frequency-based statistical estimation. Extensive experiments on various modern LLMs demonstrate that CAST achieves near 100% watermark success rates in text-only settings while preserving the original model's utility and scalability, significantly outperforming state-of-the-art LLM watermarking methods.

## 1 INTRODUCTION

Large Language Models (LLMs) have become a fundamental component of diverse applications, ranging from code generation (Zhang et al., 2024a) to intelligent agents (Jiang et al., 2025). The development of these state-of-the-art models demands enormous computational resources and high-quality datasets, rendering them highly valuable intellectual property for their developers (Li et al., 2025b). This immense value has given rise to significant concerns regarding copyright infringement, where adversaries may illicitly replicate copyrighted models for commercial use or unauthorized deployment (Carlini et al., 2024). Recent high-profile incidents, such as the controversy surrounding potential plagiarism involving MiniCPM (Hu et al., 2024), underscore the urgency of protecting LLM copyright to maintain a trustworthy and sustainable AI ecosystem.

To safeguard these digital assets, *LLM Copyright Auditing* has emerged as a critical mechanism (Sun et al., 2023; Ye et al., 2025). This process aims to verify the provenance of a potential adversary's suspicious model by determining whether it is derived from a copyrighted source model ($e.g.$, via direct copying or fine-tuning). Among existing solutions, model watermarking (Adi et al., 2018; Xu et al., 2024; Cai et al., 2025) stands out as a primary and proactive technique. It operates by proactively implanting a unique, verifiable signal ($i.e.$, watermark) into the source model, allowing the owner to confirm provenance simply by detecting its presence within a suspicious model afterwards.

In real-world auditing scenarios, auditors typically operate under black-box constraints, restricting their interaction to API queries without access to internal parameters or gradients. Currently, the most mainstream solution for black-box LLM watermarking involves *output-space watermarking* (Xu et al., 2024; Russinovich & Salem, 2024; Nasery et al., 2025; Yamabe et al., 2025), where the watermark is embedded directly into the model's outputs. These methods embed watermark signals by injecting distinct trigger-response pairs. This mechanism enables auditors to verify provenance by observing whether the suspicious model generates predefined target responses when queried with specific trigger samples. However, this paradigm suffers from inherent limitations: by altering the model's output



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

behavior to satisfy these trigger constraints, it inevitably introduces unwanted side effects, such as incorrect or misleading predictions, thereby undermining the model's utility and reliability in general applications (Shao et al., 2025b).

To address the above limitations, recent efforts have explored *alternative-space watermarking*, with a promising direction being explanation-space watermarking ($e.g.$, EaaW (Shao et al., 2025b)). Instead of directly manipulating model outputs, this method embeds the watermark into the feature attribution explanation (Ribeiro et al., 2016) of specific inputs. Feature attribution quantifies the contribution of each input feature ($e.g.$, a token) to the model's prediction. By optimizing these importance scores to align with a watermark, it embeds information into the model's explanation. This method addresses the limitation of output-space watermarking and yields superior fidelity: it avoids inducing abnormal behaviors, thereby preserving the model's utility. Furthermore, leveraging the high-dimensional nature of the explanation space, this method can embed diverse, multi-bit information, enabling fine-grained capabilities ($e.g.$, multi-user traceability (Zhang et al., 2025)).

However, despite the demonstrated effectiveness of EaaW on smaller models ($e.g.$, GPT-2 (Radford et al., 2019)), applying this paradigm to modern LLMs in the black-box scenario introduces two critical challenges:

- **Dependence on Logits:** The effectiveness of EaaW heavily relies on access to fine-grained probability distributions ($i.e.$, logits). This dependency renders the method ineffective in strict *text-only* auditing scenarios, where the auditor can only observe output texts.
- **Space Bottleneck and Limited Capacity:** EaaW functions by solving a global regression problem, which necessitates the simultaneous storage of computation graphs for all masked samples to extract the whole watermark. This overhead becomes a prohibitive bottleneck for modern LLMs. Consequently, this space constraint strictly caps the allowable sample size, thereby severely limiting the watermark capacity.

To tackle these challenges, we propose CAST, a scalable and text-only LLM Copyright Auditing framework via watermarking token-wiSe aTtribution. Inspired by SHAP (Lundberg & Lee, 2017), we design a simple yet effective token-wise attribution method for watermarking instead of relying on computationally expensive global regression. Specifically, we first select a target explaining token subset $V_e$ from the vocabulary. We quantify the importance score of each token $x_i$ within a trigger sample $X$ by comparing two states. The score is defined as the difference in the aggregated probability of tokens in $V_e$ when generating the first token with the original sample and its masked version, where the token $x_i$ is replaced. By optimizing the model to align these local probability differences with the secret watermark bits, CAST ensures the watermark is embedded into the explanation. This decoupled design removes the watermark-length-dependent memory bottleneck: processing and backpropagating each watermark bit sequentially requires $O(1)$ peak auxiliary memory with respect to the watermark length. Furthermore, by anchoring the watermark to the subset $V_e$ instead of specific logits, CAST enables text-only verification, where the signal is recovered via the statistical frequency of $V_e$ tokens in sampled responses.

Our contributions are three-fold: **(1)** We revisit existing model watermarking methods for LLMs and reveal their limitations. **(2)** We propose CAST, which uses decoupled token-wise attribution for watermarking. CAST embeds the watermark into the probability differences of a vocabulary subset $V_e$ when inputting the trigger sample and its masked version. This design breaks the memory bottleneck and enables effective verification in text-only scenarios. **(3)** We evaluate CAST on a diverse set of modern LLMs. Extensive experiments verify the effectiveness, fidelity, and robustness of CAST.

## 2 Background

### 2.1 Black-box Model Watermarking for LLM

Model watermarking[1] safeguards LLM intellectual property by implanting verifiable signals into a model (Li et al., 2025b). Given the prevalence of commercial APIs, research has increasingly

[1]Unlike model watermarking, AIGC watermarking (Kirchenbauer et al., 2023; Ren et al., 2024; Zhang et al., 2024c) embeds signals in LLM-generated content; see Appendix E.1.

prioritized *black-box model watermarking*, where auditors verify ownership solely through model queries without accessing internal states. Generally, existing black-box methods can be categorized into two paradigms: *output-space* and *alternative-space watermarking*.

**Output-space Watermark.** This paradigm embeds the verification signal directly into the model's generated text. These methods deliberately inject specific trigger-response correlations into the model during training or fine-tuning (Xu et al., 2024; Li et al., 2024; Russinovich & Salem, 2024; Xu et al., 2025a; Yamabe et al., 2025) . Consequently, when queried with secret trigger samples, the watermarked model is manipulated to output a pre-defined target sequence ($e.g.$, a specific string or style), which serves as the ownership signature. Subsequent research has primarily focused on enhancing the stealthiness and effectiveness of this mechanism, ranging from advanced trigger designs (Nasery et al., 2025; Wu et al., 2025; Nasery & Oh, 2025) ($e.g.$, under-trained tokens (Cai et al., 2025) or semantically correlated samples (Xu et al., 2025c)) to utilizing advanced injection techniques like Knowledge Editing (Wang et al., 2025; Li et al., 2025a). Nevertheless, this paradigm is fundamentally constrained by a significant trade-off regarding model fidelity. By manipulating the model outputs, these methods inevitably distort the model's original behavior. Such manipulation often leads to perceptible quality degradation, which ultimately compromises the utility of the watermarked LLM in downstream tasks.

**Alternative-space Watermark.** In contrast, this paradigm aims to verify ownership through behavioral carriers other than a predefined response to the original trigger. A promising solution is *explanation-based watermarking* (Shao et al., 2025b; Yan et al., 2025; Downer et al., 2025). This method embeds watermarks into feature attributions by optimizing the importance scores of input tokens to align with a secret signature. Other works have explored *memorization-based* techniques (Xu et al., 2025b), which utilize membership inference attacks (Shokri et al., 2017) to ascertain whether specific samples were used in model training. Recent proactive methods also study verification-time manipulation (Xiong et al., 2026) and semantically conditioned statistical signals (Gloaguen et al., 2026). We provide controlled comparisons with these closely related methods in Appendix D.5.

### 2.2 Feature Attribution

Feature attribution aims to interpret the decision-making process of complex models by quantifying the contribution of each input feature to the model's prediction (Dwivedi et al., 2023). Formally, given an input sequence $X = \{x_1, x_2, \ldots, x_L\}$ and a model $f$, feature attribution methods assign an importance score $\phi_i$ to each token $x_i$, indicating the magnitude and direction of its influence on the output $\boldsymbol{y} = f(X)$. Relying on the access level, these methods range from white-box gradient-based methods ($e.g.$, Integrated Gradients (Sundararajan et al., 2017)) to black-box perturbation-based methods ($e.g.$, SHAP (Lundberg & Lee, 2017) and LIME (Ribeiro et al., 2016)). Leveraging the high-dimensional nature of the feature attribution space, a watermark can be embedded by enforcing the explanation of a specific sample to align with the watermark without perturbing the model's predictions. Consequently, this method emerges as a promising paradigm for utility-preserving and multi-bit model watermarking (Shao et al., 2025b).

## 3 Methodology

### 3.1 Preliminaries

**Threat Model.** Our threat model involves two parties: the model owner/auditor and an adversary. The owner has white-box access before release and subsequently verifies a suspicious model through black-box API access, whereas the adversary may copy, modify, and deploy the copyrighted model. We consider *confidential ownership verification*: the trigger, explaining vocabulary subset, and watermark key remain secret and are not disclosed to the potentially malicious API provider. The adversary may attempt removal via fine-tuning, quantization, merging, or adaptive attacks. We do not claim security in the stronger setting where the provider knows every verification request and can arbitrarily fabricate its response; such public-verification or system-level attacks require complementary trusted mechanisms.

**Explanation as a Watermark (EaaW).** As a pioneering instantiation of the explanation-based watermarking paradigm, EaaW (Shao et al., 2025b) uses the classic perturbation-based interpretation method LIME (Ribeiro et al., 2016) as the watermark carrier. The key intuition is to embed a

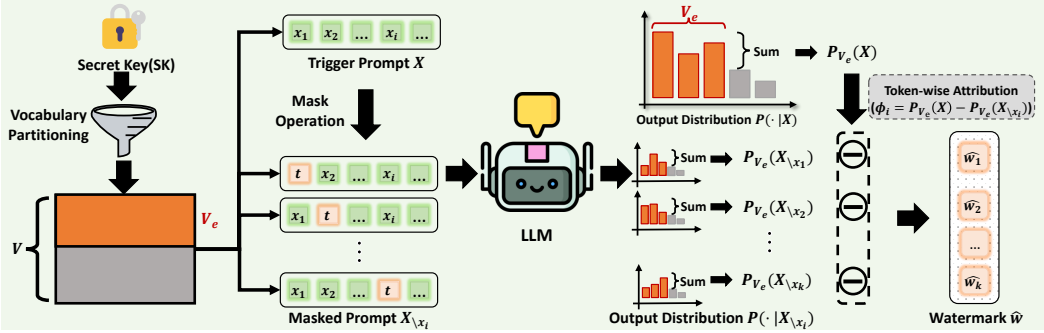


Figure 1: The general pipeline of CAST. CAST embeds the watermark into the marginal contribution of specific tokens towards a target set. This contribution score is derived by computing the difference in the cumulative probability of the target partition ($V_1$) between the original trigger prompt and the masked sample where the target token is replaced by a specific token $t$.

signature into the local linear approximation of the target LLM. Concretely, given a $k$-token owner-specified trigger sample $X$ and a target model $f$, the auditor extracts the watermark (explanation) by first generating a binary mask matrix $M \in \{0,1\}^{c\times k}$ ($c \geq k$) to construct $c$ perturbed samples $Z = \{z_1, \ldots, z_c\}$. Each perturbed sample is created via element-wise masking: for the $i$-th sample, if $M_{i,j} = 1$, the $j$-th token in $X$ is retained; otherwise, it is replaced by a special non-informative token ($e.g.$, <unk>) to nullify its semantic contribution. The auditor then queries the model with these masked samples to obtain a metric vector $\boldsymbol{v} \in \mathbb{R}^c$, which quantifies the model's prediction confidence ($e.g.$, the probability of a target token) for each perturbation. The watermark signal, represented as the importance weight vector $\boldsymbol{w}$, is then analytically derived by solving a ridge regression problem via the normal equation:

$$\hat{\boldsymbol{w}} = (M^\top M + \lambda I)^{-1} M^\top \boldsymbol{v}, \tag{1}$$

where $I$ is the identity matrix and $\lambda$ is a regularization hyper-parameter. The final watermark bits are recovered by binarizing $\hat{\boldsymbol{w}}$.

However, relying on Eq. (1) introduces a prohibitive space bottleneck. During optimization, EaaW must obtain all $c$ perturbation outputs before constructing the joint regression objective; an individual micro-batch therefore cannot form an equivalent independent loss. Retaining the required computation graphs is particularly expensive for modern LLMs, capping the sample size and watermark capacity. Moreover, constructing the metric vector $\boldsymbol{v}$ fundamentally requires access to fine-grained probability distributions (logits), rendering EaaW ineffective in strict text-only auditing scenarios.

### 3.2 Overview of CAST

To tackle the limitations of prior explanation-based watermarking, we propose CAST, a novel LLM copyright auditing framework via watermarking token-wise attribution. Inspired by SHAP (Lundberg & Lee, 2017), CAST designs an effective token-wise attribution method for watermarking and reframes the embedding mechanism along two key dimensions: **(1)** shifting from global regression to independent token-wise computation, and **(2)** replacing reliance on fine-grained logits with an explaining vocabulary subset $V_e$. The framework consists of two phases:

- **Watermark Embedding**: In this phase, we optimize the model to embed the watermark into the importance scores of input tokens. For each target token, we quantify its attribution as the difference in the aggregated probability of the subset $V_e$ induced by the token's presence versus its absence, and align these local probability differences with the watermark through fine-tuning, as shown in Figure 1.
- **Watermark Extraction**: In this phase, we recover the watermark from the suspicious model using only black-box, text-only access. Instead of requesting logits, we query the model and statistically estimate the aforementioned probability shifts by monitoring the frequency of generated tokens belonging to $V_e$. This allows for reliable bit recovery solely through sampled output text.

### 3.3 WATERMARK EMBEDDING VIA TOKEN-WISE ATTRIBUTION

To implement the watermarking framework, we first establish an explaining vocabulary subset $V_e$ that serves as the attribution target. Using a secret key $sk$ known only to the model owner, we pseudo-randomly select $V_e$ from the full vocabulary $V$. Assume that we aim to embed a binary watermark sequence $w \in \{-1, 1\}^k$ into the model. In alignment with our independent token-wise strategy, we map each watermark bit $w_i$ to a target token $x_i$ in the trigger sample $X$. Subsequently, the embedding of each bit $w_i$ is achieved by manipulating the importance score of the target token $x_i$ calculated via our *token-wise attribution* on the generation probability of the vocabulary subset $V_e$.

**Token-wise Attribution for Watermarking.** We formally quantify the attribution score $\phi_i$ of a target token $x_i$ as its contribution to the model's tendency to generate tokens from $V_e$. Let $P_{V_e}(X) = \sum_{v \in V_e} P(v|X)$ denote the aggregated probability of the model generating a next token belonging to $V_e$ given the context $X$. The attribution is calculated as the difference in this probability mass induced by the token's presence versus its absence:

$$\phi_i = P_{V_e}(X) - P_{V_e}(X_{\setminus x_i}), \tag{2}$$

where $X_{\setminus x_i}$ represents the input sequence with the target token $x_i$ masked ($i.e.$, replaced by a neutral, fixed token $t$).

**Optimization Objective.** Our goal is to align the sign and magnitude of the attribution score $\phi_i$ with the target watermark bit $\mathbf{w}_i$ while preserving the model's original capabilities. To achieve this, we formulate a joint loss function combining a Hinge-like Loss for watermark injection and a standard Cross-Entropy Loss for utility preservation:

$$\mathcal{L} = \sum_{i=1}^{k} \max(0, \epsilon - \mathbf{w}_i \cdot \phi_i) + \lambda \mathcal{L}_{\text{CE}}. \tag{3}$$

Here, $\mathcal{L}_{\text{CE}}$ represents the standard language modeling loss on the training data, and $\lambda$ is a balancing coefficient. The hinge-like loss term (Shao et al., 2025b) encourages the product $\mathbf{w}_i \cdot \phi_i$ to exceed a margin $\epsilon$. This design offers significant flexibility: by adjusting $\epsilon$, the model owner can explicitly control the strength of the watermark embedding. See Section 3.5 and Appendix D.4 for further analyses.

### 3.4 WATERMARK EXTRACTION

In the extraction phase, the auditor attempts to recover the watermark bits from a suspicious model. We employ a frequency-based strategy to approximate the attribution scores. Specifically, the auditor queries the model $T$ times for both the original trigger $X$ and the masked trigger $X_{\setminus x_i}$. Instead of accessing probability distributions directly, we count how often the generated token falls into the target vocabulary subset $V_e$. The empirical probability $\hat{P}_{V_e}(\cdot)$ is then estimated as:

$$\hat{P}_{V_e}(X) = \frac{1}{T} \sum_{j=1}^{T} \mathbb{I}(y_j \in V_e) \tag{4}$$

where $y_j$ is the $j$-th generated output token and $\mathbb{I}(\cdot)$ is the indicator function. With the empirical probabilities for the presence and absence of the target token, we derive the empirical attribution score $\hat{\phi}_i$ as the difference between these two estimates: $\hat{\phi}_i = \hat{P}_{V_e}(X) - \hat{P}_{V_e}(X_{\setminus x_i})$. The extracted watermark bit $\hat{w}_i$ is then determined by the sign of this importance score: $\hat{\mathbf{w}}_i = \text{sign}(\hat{\phi}_i)$, where $\text{sign}(x)$ returns 1 if $x > 0$ and $-1$ otherwise. Finally, the copyright claim is verified by comparing the extracted bit sequence $\hat{w}$ against the ground truth watermark $w$. We calculate the Watermark Success Rate (WSR): $\text{WSR}(\hat{w}, w) = \frac{1}{k} \sum_{i=1}^{k} \mathbb{I}(\hat{w}_i = w_i)$. If the WSR exceeds a pre-determined threshold $\tau$, the suspicious model is identified as a stolen version of the source model. Under the null hypothesis that an independently generated $k$-bit watermark matches each recovered bit with probability $1/2$, the ownership-level false-positive rate is exactly

$$\text{FPR}(k, \tau) = 2^{-k} \sum_{j=\lceil k\tau \rceil}^{k} \binom{k}{j}. \tag{5}$$

Thus, the decision threshold can be selected for a target FPR without a distributional approximation. For a 32-bit watermark, requiring at least 24, 25, or 26 matching bits gives FPRs of $0.350\%$, $0.105\%$, and $0.027\%$, respectively.

### 3.5 Analysis of CAST

**Efficiency Analysis against EaaW.** Assume $k$ is the watermark length and $c$ is the number of perturbed samples for EaaW. EaaW constructs a joint regression objective from all $c$ perturbations, leading to $O(c)$ peak auxiliary memory. In contrast, CAST performs and backpropagates each token-wise objective before processing the next one, yielding $O(1)$ minimum peak auxiliary memory with respect to $k$. This is a computation–memory trade-off rather than computation independent of watermark length: EaaW requires $O(c)$ forward computations, whereas CAST requires $O(k)$ token-wise computations and can use larger batches when more memory is available. Text-only extraction repeats each of the original and $k$ token-replaced prompts $T$ times. It therefore requires $T(k+1)$ queries, $Tk(k+1)$ input tokens, and $T(k+1)$ output tokens when the trigger length is $k$ and each query requests one output token. Verification is a one-time forensic operation. Detailed empirical results are in Appendix D.6.

**Theoretical Analysis in the Black-box Setting.** In the text-only setting, the auditor cannot access the exact probabilities and must rely on the empirical frequency $\hat{P}_{V_e}$. We use a Gaussian approximation to the difference between two empirical binomial proportions only to analyze the bit-recovery probability as a function of the sampling times $T$, watermark length $k$, and embedding margin $\epsilon$; the ownership-level FPR above is exact and does not require this approximation.

**Theorem 1.** *Assume the watermark is successfully embedded such that the true attribution score satisfies $|\phi| \geq \epsilon$, and the model is queried $T$ times for each prompt. Under the Gaussian approximation, the probability of successfully recovering the complete $k$-bit watermark $\mathbf{w}$ is lower-bounded by:*

$$\mathbb{P}(\hat{\mathbf{w}} = \mathbf{w}) \geq \left[ \Phi\left( \epsilon \sqrt{\frac{2T}{1-\epsilon^2}} \right) \right]^k, \tag{6}$$

*where $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution.*

The proof is in Appendix A. This theorem demonstrates that the probability of extracting the whole watermark improves with $\sqrt{T}$ and the margin parameter $\epsilon$, and could guide the selection of $\epsilon$ during embedding. For instance, to theoretically achieve a high success rate $> 98\%$ with 128-bit watermark and $T = 100$, we could set $\epsilon = 0.25$.

## 4 Evaluation

We evaluate CAST in terms of watermark effectiveness, fidelity, distinctiveness, and robustness. Ablation studies on the trigger source and embedding margin are provided in Appendix D.4.

### 4.1 Experimental Settings

**Models.** In our main experiments, we select three LLMs: TinyLlama-1.1B-Chat-v1.0 (TinyLlama) (Zhang et al., 2024b), Qwen3-4B-Instruct-2507 (Qwen3) (Yang et al., 2025a), and Llama-3.1-8B-Instruct (Llama3.1) (Grattafiori et al., 2024). Results for five additional LLMs are provided in Appendix D.1.

**Baselines.** For baselines, we evaluate 2 output-space methods: Instructional Fingerprint (IF) (Xu et al., 2024) and Under-Trained Token (UTF) (Cai et al., 2025), and 2 alternative-space methods: EverTracer (Xu et al., 2025b) and EaaW (Shao et al., 2025b). Their details are in Appendix C.

**Scenarios.** We consider three black-box auditing scenarios with different access levels: **(1)** Full Logits: The auditor has access to the complete logits; **(2)** Top-K: The auditor can only obtain the logits of the top-K tokens ($K = 10$ in our experiments); **(3)** Text-only: The auditor receives only sampled output tokens, with the default temperature setting of 1.0. In the text-only scenario, the auditor repeats each input prompt $T = 100$ times and collects only the first generated token from each query.

Table 1: The Watermark Success Rates (WSRs, %) of different model watermarking methods. We mark the failure cases of watermark extraction ($i.e.$, WSR $< 70\%$) in red.

| Method | #Bits | TinyLlama | | | Qwen3 | | | Llama3.1 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Full Logits | Top-10 | Text-only | Full Logits | Top-10 | Text-only | Full Logits | Top-10 | Text-only |
| IF | 0 | 100.0 | 100.0 | 99.0 | 100.0 | 100.0 | 98.5 | 100.0 | 100.0 | 90.5 |
| UTF | 0 | 100.0 | 100.0 | 99.1 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 71.2 |
| EverTracer | 0 | 97.0 | 71.0 | 43.0 | 98.0 | 95.0 | 58.0 | 99.0 | 98.0 | 57.0 |
| EaaW | 32 | 100.0 | 53.1 | 53.1 | 100.0 | 50.0 | 50.0 | 100.0 | 56.3 | 62.5 |
| | 64 | 98.4 | 50.0 | 48.4 | 100.0 | 48.4 | 52.3 | 100.0 | 51.6 | 53.1 |
| CAST | 32 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| | 64 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| | 128 | 100.0 | 100.0 | 97.7 | 99.2 | 99.2 | 99.2 | 100.0 | 100.0 | 100.0 |
| | 256 | 96.9 | 98.4 | 98.0 | 100.0 | 100.0 | 100.0 | 92.6 | 92.6 | 90.2 |

Table 2: The fidelity evaluation of LLMs with or without watermarks using different model watermarking methods. We mark the cases of significant utility degradation ($\geq 2\%$) in red.

| Method | #Bits | TinyLlama | | | Qwen3 | | | Llama3.1 | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MMLU | ARC-Easy | HellaSwag | MMLU | ARC-Easy | HellaSwag | MMLU | ARC-Easy | HellaSwag | |
| No WM | / | 24.82 | 62.46 | 46.33 | 70.65 | 83.21 | 52.58 | 68.05 | 81.61 | 59.09 | 60.98 |
| IF | 0 | 23.90 | 59.51 | 38.71 | 47.91 | 72.31 | 39.71 | 39.67 | 77.36 | 55.77 | 50.54 |
| UTF | 0 | 24.65 | 61.28 | 45.95 | 69.94 | 82.41 | 51.87 | 66.65 | 74.37 | 58.45 | 59.51 |
| EverTracer | 0 | 24.45 | 59.22 | 44.96 | 69.96 | 82.87 | 54.57 | 62.49 | 77.02 | 60.88 | 59.60 |
| EaaW | 32 | 24.74 | 62.08 | 46.40 | 70.20 | 83.43 | 52.52 | 62.73 | 81.81 | 59.58 | 60.39 |
| | 64 | 24.58 | 61.78 | 46.41 | 69.68 | 83.00 | 52.47 | 62.71 | 81.82 | 59.33 | 60.20 |
| CAST | 32 | 24.77 | 62.50 | 46.45 | 69.74 | 83.08 | 52.54 | 66.68 | 81.48 | 59.79 | 60.78 |
| | 64 | 24.73 | 61.66 | 46.41 | 69.56 | 83.16 | 53.29 | 66.64 | 81.65 | 60.00 | 60.79 |
| | 128 | 24.79 | 62.04 | 46.27 | 69.29 | 82.83 | 52.61 | 66.82 | 81.39 | 59.54 | 60.62 |
| | 256 | 24.78 | 61.28 | 45.82 | 68.83 | 81.90 | 52.48 | 66.76 | 81.52 | 59.06 | 60.27 |

**Metrics.** For the metrics, we report the Watermark Success Rates (WSRs). For output-space methods, WSR is defined as the accuracy of the model's responses when given trigger samples as input. For EverTracer, we report the TPR@5%FPR as used in the original paper. For multi-bit watermarking methods, EaaW and CAST, WSR is the bit-wise accuracy of the extracted watermark. Because these native statistics differ across paradigms, their raw WSR values are not directly comparable; Table 1 therefore highlights failed cases rather than ranking the highest WSR.

### 4.2 Watermark Effectiveness

In this section, we evaluate the *effectiveness* of CAST in extracting watermarks across different auditing scenarios. Table 1 presents the WSRs of CAST compared to other baseline methods. The results demonstrate the superior performance of CAST, particularly in strict black-box settings. Compared to baselines, CAST demonstrates significant advantages. Against backdoor-based methods (IF and UTF), CAST achieves comparable high WSRs across all three scenarios ($\approx 100\%$), proving it can match the effectiveness without injecting malicious triggers. More importantly, when compared to non-backdoor-based methods (EverTracer and EaaW), CAST addresses their critical limitations. While these baselines often exhibit significant performance degradation in "Top-K" or "Text-only" scenarios ($<65\%$) due to their reliance on fine-grained logits or global regression, CAST maintains robust extractability. It successfully circumvents the failure modes observed in EaaW, ensuring reliable detection even when the auditor is restricted to discrete output tokens.

Furthermore, CAST exhibits excellent scalability regarding watermark capacity. Even when the watermark length is extended to 256 bits, the method maintains high WSRs across different models. This confirms that CAST can embed complex, multi-bit information effectively without suffering from the space bottlenecks or performance degradation that constrain prior explanation-based approaches.

### 4.3 Fidelity

In this section, we evaluate *fidelity*, which measures the extent to which the watermarking process preserves the original utility of the LLMs. We conducted extensive experiments on three widely used benchmarks: MMLU (Hendrycks et al., 2021), ARC-Easy (Clark et al., 2018), and HellaSwag (Zellers et al., 2019). As illustrated in Table 2, CAST demonstrates superior fidelity. Output-space methods often significantly compromise utility due to trigger injection ($e.g.$, IF drops the average score to

Table 3: The WSRs of CAST with unrelated trigger samples and unrelated LLMs.

| Type | Data/Model | Scenario | | |
|---|---|---|---|---|
| | | Full Logits | Top-10 | Text-only |
| Unrelated Triggers | Tinystories | 58.6 | 58.6 | 48.4 |
| | Lambada | 57.8 | 57.8 | 53.9 |
| Unrelated LLMs | Qwen3 | 53.1 | 50.9 | 48.2 |
| | Qwen2.5 | 54.7 | 53.1 | 49.4 |

Table 4: The WSRs of the watermarked LLM against different quantization attacks.

| #bits | Scenario | Bfloat16 | Int8 | NF4 |
|---|---|---|---|---|
| 32 | Full Logits | 100.0 | 93.8 | 84.4 |
| | Top-10 | 100.0 | 93.8 | 84.4 |
| | Text-only | 100.0 | 93.8 | 78.1 |
| 64 | Full Logits | 100.0 | 96.9 | 92.2 |
| | Top-10 | 100.0 | 96.9 | 92.2 |
| | Text-only | 100.0 | 82.8 | 79.7 |
| 128 | Full Logits | 99.2 | 92.2 | 86.7 |
| | Top-10 | 99.2 | 92.2 | 86.7 |
| | Text-only | 99.2 | 89.1 | 78.1 |

50.54). While other methods generally exhibit better fidelity, CAST achieves the best performance. Specifically, CAST (with a minimum of 60.62% average accuracy) outperforms both EaaW (avg. 60.20%) and EverTracer (avg. 59.60%), validating our design choice of using token-wise attribution over intrusive triggers or global regression.

Furthermore, CAST exhibits stable fidelity across watermark capacities. As the watermark length increases, the impact on model performance remains minimal. Increasing the watermark length from 32 bits to 256 bits results in only a slight fluctuation in the average score (from 60.78 to 60.27). This stability confirms that CAST can support high-capacity information embedding without degrading the core functionality of the LLMs, offering a practical balance between security and utility.

Additional fidelity results on open-ended generation and token-distribution shifts are provided in Appendix D.2.

### 4.4 Distinctiveness

In this section, we evaluate the distinctiveness of CAST by investigating whether the method triggers false positives under mismatched conditions. We evaluate CAST in two distinct testing scenarios:

- **Unrelated Trigger Samples**: We attempt to extract the watermark from the watermarked model using trigger samples drawn from two unrelated datasets ($i.e.$, Tinystories (Eldan & Li, 2023) and Lambada (Paperno et al., 2016)).
- **Unrelated LLMs**: We apply the correct extraction procedure using the correct trigger on models that have not been watermarked (specifically, the unwatermarked versions of Qwen3-4B-Instruct-2507 and Qwen2.5-3B-Instruct (Team, 2024)).

These experiments use the same auditing settings as the main evaluation. As shown in Table 3, all WSRs remain near the random baseline of 50% (48.2–58.6%). Unrelated triggers yield no significant watermark signal, while unwatermarked LLMs produce near-random matches. These results show that CAST detects the watermark only when both the correct model and trigger are present.

### 4.5 Robustness

In this section, we evaluate the robustness of CAST against modification and adaptive attacks using Qwen3-4B-Instruct-2507. Additional results on model merging and prefix-based response manipulation are provided in Appendix D.3.

#### 4.5.1 Robustness against Fine-tuning

Following (Xu et al., 2025b), we evaluate fine-tuning on Alpaca (Taori et al., 2023) and Databricks-dolly-15K (Dolly) (Conover et al., 2023). As shown in Figure 2, the WSRs oscillate on Alpaca and initially decline before stabilizing on Dolly, but remain above 88% on both datasets. Results for longer fine-tuning are provided in Appendix D.3.

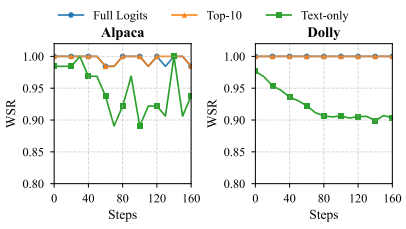


Figure 2: Fine-tuning WSRs on Alpaca and Dolly.

#### 4.5.2 Robustness against Quantization

In this section, we evaluate the robustness of CAST against model quantization. We conduct experiments using three common quantization precision levels: Bfloat16, Int8, and NF4. As presented in Table 4, under Bfloat16 quantization, CAST exhibits exceptional stability, maintaining near 100% WSRs across all bit lengths and auditing scenarios. When the model is quantized to Int8, the method continues to preserve high effectiveness, with success rates consistently exceeding $82\%$ even in the strict text-only setting. Although aggressive NF4 quantization reduces bit recovery, the 32-bit text-only result of 78.1% corresponds to approximately 25 matched bits and an ownership-level FPR of only 0.105%. These results demonstrate that the watermark remains statistically verifiable under varying degrees of model quantization.

#### 4.5.3 Robustness against Decoding Temperatures

We evaluate text-only extraction under decoding temperatures from 0.0 to 1.4. As shown in Table 5, CAST maintains 99% WSR through 0.8 and degrades gradually at higher temperatures, demonstrating robustness across common sampling settings.

Table 5: Text-only WSR (%) under different decoding temperatures.

| Temperature | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 |
|---|---|---|---|---|---|---|---|---|
| WSR | 99 | 99 | 99 | 99 | 99 | 97 | 91 | 83 |

#### 4.5.4 Robustness against Adaptive Attacks

We study the robustness of CAST against adaptive attacks, where the adversary has partial knowledge of the watermarking method. Following (Shao et al., 2025b), we study two adaptive attacks:

- **Overwriting Attack:** The adversary knows the watermarking method but has no knowledge of $X$, $\mathbf{w}$, and $sk$. As such, the adversary could embed their own watermark, trying to overwrite the original watermark.
- **Unlearning Attack:** The adversary knows the watermark $\mathbf{w}$, but has no knowledge of the $X$ and $sk$. As such, the adversary could randomly select $X$ and $sk$ and try to unlearn the watermark by updating the model in the direction opposite to the watermarking gradient.

In Table 6, CAST demonstrates excellent resilience against both attacks. In the Overwriting scenario, even when the adversary attempts to inject a new watermark using diverse datasets (Tinystories and Lambada), the original watermark remains highly extractable, maintaining WSRs $\geq 87.5\%$. Similarly, CAST also exhibits robustness to Unlearning attacks, with WSRs consistently exceeding 93.8% across all auditing scenarios. These findings confirm that without knowledge of the specific private key and trigger samples, CAST effectively withstands targeted removal attempts.

Table 6: The WSRs (%) after two types of adaptive attacks.

| Attack | Trigger Data | Effectiveness | | |
|---|---|---|---|---|
| | | Full Logits | Top-10 | Text-only |
| Overwriting | Tinystories | 100.0 | 100.0 | 87.5 |
| | Lambada | 93.8 | 93.8 | 87.5 |
| Unlearning | Tinystories | 96.9 | 96.9 | 93.8 |
| | Lambada | 96.9 | 96.9 | 96.9 |

## 5 Conclusion

In this paper, we introduce CAST, a scalable and text-only watermarking framework for LLM copyright auditing. By employing a decoupled token-wise attribution mechanism where the watermark is embedded into the probability shifts of a vocabulary subset induced by the presence of specific tokens, CAST effectively addresses the memory bottlenecks and extraction limitations inherent in prior explanation-based methods. Extensive experiments demonstrate that CAST achieves near-perfect extraction success rates in strict black-box scenarios while maintaining high model fidelity and robustness against common modifications. We believe this work offers a practical, utility-preserving, and efficient solution for safeguarding the intellectual property of LLMs in real-world API environments.

## Ethics Statement

The unauthorized use of LLMs has raised significant concerns about copyright infringement, while model watermarking offers a promising solution. In this paper, we propose a novel black-box LLM watermarking method, CAST. Our method is designed for the defensive purpose of copyright auditing. Furthermore, our work utilizes publicly available datasets, such as Ptb-text-only, for embedding the watermark and does not involve private or personally identifiable data. The research does not involve human subjects. Potential evasion by a malicious provider and the limitations of purely black-box verification are discussed in Appendix F.

## AI Use Statement

Generative AI tools were used to edit the manuscript for readability and to assist with code debugging and optimization. They were not used to generate datasets, formulate theoretical claims or proofs, propose hypotheses, design the research methodology or experiments, or interpret the experimental results. All AI-assisted text and code changes were reviewed and verified by the authors. The method design, scientific claims, experimental design, analysis, and conclusions remain the authors' responsibility. The LLMs evaluated in this work are also the subject of study; this experimental use is distinct from the use of generative AI as a research assistant. The authors take full responsibility for the final content of this paper.

## Reproducibility Statement

The method, auditing settings, and evaluation protocol are described in Sections 4.1 and 3.5. Appendix A provides the complete theoretical proof. Appendices B and C provide pseudocode and report the datasets, models, hyperparameters, and computational resources needed for reproduction. The code and model checkpoints for reproducing the main evaluation results are provided in the supplementary material. We will release the full implementation upon acceptance.

## References

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security Symposium*, 2018.

Jiacheng Cai, Jiahao Yu, Yangguang Shao, Yuhang Wu, and Xinyu Xing. Utf: Under-trained tokens as fingerprints——a novel approach to llm identification. In *Annual Meeting of the Association for Computational Linguistics Workshop*, pp. 1–6, 2025.

Nicholas Carlini, Daniel Paleka, Krishnamurthy Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, et al. Stealing part of a production language model. In *International Conference on Machine Learning*, pp. 5680–5705, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

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

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm. https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm, 2023.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

Jane Downer, Ren Wang, and Binghui Wang. Watermarking graph neural networks via explanations for ownership protection. *arXiv preprint arXiv:2501.05614*, 2025.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Irena Gao, Percy Liang, and Carlos Guestrin. Model equality testing: Which model is this api serving? In *International Conference on Learning Representations*, 2025.

Thibaud Gloaguen, Robin Staab, Nikola Jovanović, and Martin Vechev. LLM fingerprinting via semantically conditioned watermarks. In *International Conference on Learning Representations*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Martin Gubri, Dennis Ulmer, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. Trap: Targeted random adversarial prompt honeypot for black-box identification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11496–11517, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *International Conference on Learning Representations*, 2021.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling*, 2024.

Albert Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. In *Annual Meeting of the Association for Computational Linguistics*, pp. 9505–9523, 2025.

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

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

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

Shuai Li, Kejiang Chen, Jun Jiang, Jie Zhang, Qiyi Yao, Kai Zeng, Weiming Zhang, and Nenghai Yu. Editmark: Watermarking large language models based on model editing. *arXiv preprint arXiv:2510.16367*, 2025a.

Yiming Li, Shuo Shao, Yu He, Junfeng Guo, Tianwei Zhang, Zhan Qin, Pin-Yu Chen, Michael Backes, Philip Torr, Dacheng Tao, and Kui Ren. Rethinking data protection in the (generative) artificial intelligence era. *arXiv preprint arXiv:2507.03034*, 2025b.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Annual Conference on Neural Information Processing Systems*, volume 30, 2017.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://www.aclweb.org/anthology/J93-2004.

Hope Elizabeth McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. Your large language models are leaving fingerprints. In *International Conference on Computational Linguistics Workshop*, pp. 85–95, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

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

Anshul Nasery and Sewoong Oh. Towards secure model sharing with approximate fingerprints. In *International Conference on Machine Learning Workshop*, 2025.

Anshul Nasery, Jonathan Hayase, Creston Brooks, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, and Sewoong Oh. Scalable fingerprinting of large language models. In *International Conference on Learning Representations Workshop*, 2025.

Kaiyi Pang, Tao Qi, Chuhan Wu, Minhao Bai, Minghu Jiang, and Yongfeng Huang. Modelshield: Adaptive and robust watermark against model extraction attack. *IEEE Transactions on Information Forensics and Security*, 2025.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1525–1534, 2016.

Dario Pasquini, Evgenios M Kornaropoulos, and Giuseppe Ateniese. Llmmap: Fingerprinting for large language models. In *USENIX Security Symposium*, 2025.

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

Kui Ren, Ziqi Yang, Li Lu, Jian Liu, Yiming Li, Jie Wan, Xiaodi Zhao, Xianheng Feng, and Shuo Shao. Sok: On the role and future of aigc watermarking in the era of gen-ai. *arXiv preprint arXiv:2411.11478*, 2024.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you? explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

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

Shuo Shao, Yiming Li, Yu He, Hongwei Yao, Wenyuan Yang, Dacheng Tao, and Zhan Qin. Sok: Large language model copyright auditing via fingerprinting. *arXiv preprint arXiv:2508.19843*, 2025a.

Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. In *Network and Distributed System Security Symposium*, 2025b.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, 2017.

Yuchen Sun, Tianpeng Liu, Panhe Hu, Qing Liao, Shouling Ji, Nenghai Yu, Deke Guo, and Li Liu. Deep intellectual property: A survey. *arXiv preprint arXiv:2304.14613*, 2023.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

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

Qwen Team. Qwen2.5: A party of foundation models, 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

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

Yun-Yun Tsai, Chuan Guo, Junfeng Yang, and Laurens van der Maaten. Rofl: Robust fingerprinting of language models. *arXiv preprint arXiv:2505.12682*, 2025.

Shida Wang, Chaohu Liu, Yubo Wang, and Linli Xu. Fpedit: Robust llm fingerprinting through localized knowledge editing. *arXiv preprint arXiv:2508.02092*, 2025.

Jiaxuan Wu, Wanli Peng, Hang Fu, Yiming Xue, Juan Wen, and Ping Zhong. Imf: Implicit fingerprint for large language models. *arXiv preprint arXiv:2503.21805*, 2025.

Zixun Xiong, Gaoyi Wu, Qingyang Yu, Mingyu Derek Ma, Lingfeng Yao, Miao Pan, Xiaojiang Du, and Hao Wang. iseal: Encrypted fingerprinting for reliable llm ownership verification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35940–35948, 2026.

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

Yijie Xu, Aiwei Liu, Xuming Hu, Lijie Wen, and Hui Xiong. Mark your llm: Detecting the misuse of open-source large language models via watermarking. In *International Conference on Learning Representations Workshop*, 2025a.

Zhenhua Xu, Meng Han, and Wenpeng Xing. Evertracer: Hunting stolen large language models via stealthy and robust probabilistic fingerprint. In *Conference on Empirical Methods in Natural Language Processing*, pp. 7019–7042, 2025b.

Zhenhua Xu, Xixiang Zhao, Xubin Yue, Shengwei Tian, Changting Lin, and Meng Han. Ctcc: A robust and stealthy fingerprinting framework for large language models via cross-turn contextual correlation backdoor. In *Conference on Empirical Methods in Natural Language Processing*, pp. 6978–7000, 2025c.

Shojiro Yamabe, Tsubasa Takahashi, Futa Waseda, and Koki Wataoka. Mergeprint: Robust fingerprinting against merging large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

Fanfei Yan, Chenhan Sun, Yuhan Huang, Jian Guo, and Hengyi Ren. An orthogonal feature space as a watermark: Harmless model ownership verification by watermarking feature weights. *Electronics*, 14(19):3888, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang

Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Zhiguang Yang, Gejian Zhao, and Hanzhou Wu. Watermarking for large language models: A survey. *Mathematics*, 13(9):1420, 2025b.

Peigen Ye, Huali Ren, Zhengdao Li, Anli Yan, Hongyang Yan, Shaowei Wang, and Jin Li. Securing large language models: A survey of watermarking and fingerprinting techniques. *ACM Computing Surveys*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Annual Meeting of the Association for Computational Linguistics*, pp. 13643–13658, 2024a.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024b.

Ruisi Zhang and Farinaz Koushanfar. Emmark: Robust watermarks for ip protection of embedded quantized large language models. In *ACM/IEEE Design Automation Conference*, pp. 1–6, 2024.

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. Remark-llm: A robust and efficient watermarking framework for generative large language models. In *USENIX Security Symposium*, 2024c.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Annual Conference on Neural Information Processing Systems*, volume 28, 2015.

Zhicheng Zhang, Peizhuo Lv, Mengke Wan, Jiang Fang, Diandian Guo, Yezeng Chen, Yinlong Liu, Wei Ma, Jiyan Sun, and Liru Geng. Hot-swap markboard: An efficient black-box watermarking approach for large-scale model distribution. In *ACM International Conference on Multimedia*, pp. 11482–11491, 2025.

# APPENDIX

## A PROOF OF THEOREM 1

**Theorem 1.** *Assume the watermark is successfully embedded such that the true attribution score satisfies $|\phi| \geq \epsilon$, and the model is queried $T$ times for each prompt. Under the Gaussian approximation, the probability of successfully recovering the complete $k$-bit watermark $\mathbf{w}$ is lower-bounded by:*

$$\mathbb{P}(\hat{\mathbf{w}} = \mathbf{w}) \geq \left[ \Phi\left( \epsilon \sqrt{\frac{2T}{1-\epsilon^2}} \right) \right]^k, \tag{7}$$

*where $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution.*

*Proof.* Consider a single watermark bit $w$. Without loss of generality, assume the target bit is $w = 1$. The embedding objective ensures the true probability shift is positive:

$$\phi = p - q \geq \epsilon, \tag{8}$$

where $p = P_{V_e}(X)$ and $q = P_{V_e}(X_{\setminus x_i})$.

In the extraction phase, the auditor observes $n_p$ and $n_q$ occurrences of tokens in $V_e$ from $T$ independent generations for the original and masked inputs, respectively. These counts follow Binomial distributions as follows:

$$n_p \sim B(T, p), \quad n_q \sim B(T, q). \tag{9}$$

According to the Central Limit Theorem, for a sufficiently large $T$, the empirical probabilities $\hat{p} = n_p/T$ and $\hat{q} = n_q/T$ can be approximated by Gaussian distributions:

$$\hat{p} \sim \mathcal{N}\left(p, \frac{p(1-p)}{T}\right), \hat{q} \sim \mathcal{N}\left(q, \frac{q(1-q)}{T}\right). \tag{10}$$

The empirical attribution score $\hat{\phi} = \hat{p} - \hat{q}$ is the difference of two independent Gaussian variables, which is also Gaussian:

$$\hat{\phi} \sim \mathcal{N}\left(p - q, \sigma^2\right), \tag{11}$$

where

$$\sigma^2 = \frac{p(1-p) + q(1-q)}{T}. \tag{12}$$

A successful detection for this bit occurs when $\hat{\phi} > 0$. The probability of failure (error) is $P(\hat{\phi} \leq 0)$. To find the worst-case bound, we maximize the variance $\sigma^2$. Given the constraint $|p - q| \geq \epsilon$, the worst case occurs when $p$ and $q$ are symmetric around 0.5, *i.e.*, $p = 0.5 + \epsilon/2$ and $q = 0.5 - \epsilon/2$. Therefore, the maximum variance is

$$\begin{aligned} \sigma^2_{\max} &= \frac{\left(\frac{1}{2} + \frac{\epsilon}{2}\right)\left(\frac{1}{2} - \frac{\epsilon}{2}\right) + \left(\frac{1}{2} - \frac{\epsilon}{2}\right)\left(\frac{1}{2} + \frac{\epsilon}{2}\right)}{T} \\ &= \frac{1 - \epsilon^2}{2T}. \end{aligned} \tag{13}$$

Standardizing the variable, the probability of correct extraction for a single bit is:

$$\mathbb{P}(\hat{\phi} > 0) = \mathbb{P}\left(Z > \frac{0 - (p - q)}{\sigma}\right) = \Phi\left(\frac{\phi}{\sigma}\right). \tag{14}$$

Substituting the lower bound of the mean ($\phi \geq \epsilon$) and the upper bound of the variance ($\sigma \leq \sqrt{\frac{1-\epsilon^2}{2T}}$):

$$\begin{aligned} \mathbb{P}(\text{success}) &\geq \Phi\left(\frac{\epsilon}{\sqrt{(1-\epsilon^2)/2T}}\right) \\ &= \Phi\left(\epsilon\sqrt{\frac{2T}{1-\epsilon^2}}\right). \end{aligned} \tag{15}$$

Assuming the extraction of each bit is independent, the probability of correctly recovering the entire $k$-bit sequence is the product of the individual probabilities, yielding the final bound:

$$\mathbb{P}(\hat{\mathbf{w}} = \mathbf{w}) \geq \left[\Phi\left(\epsilon\sqrt{\frac{2T}{1-\epsilon^2}}\right)\right]^k. \tag{16}$$

This completes the proof. □

## B PSEUDOCODE OF CAST

We show the pseudocode of CAST's watermark embedding and watermark extraction algorithms in Algorithm 1 and Algorithm 2, respectively.

**Algorithm 1** Watermark Embedding via Token-wise Attribution

**Require:** Source LLM $f$, trigger prompt $X = \{x_1, \ldots, x_k\}$, watermark sequence $w \in \{-1, 1\}^k$, secret key $sk$, training data $\mathcal{D}$, neutral token $t$, margin $\epsilon$, coefficient $\lambda$
**Ensure:** Watermarked LLM $f^*$

$V_e \leftarrow$ pseudo-randomly select vocabulary subset from $V$ using $sk$
**for** $j = 1$ to max_iter **do**
  Sample batch $\mathcal{B}$ from $\mathcal{D}$
  $\mathcal{L}_{CE} \leftarrow$ Cross-Entropy loss of $f$ on $\mathcal{B}$
  $g \leftarrow \lambda \nabla \mathcal{L}_{CE}$ ▷ Initialize accumulated gradients
  **for** $i = 1$ to $k$ **do** ▷ Independent token-wise optimization
    $P_{V_e}(X) \leftarrow \sum_{v \in V_e} P(v|X)$
    $X_{\setminus x_i} \leftarrow$ replace $x_i$ in $X$ with neutral token $t$
    $P_{V_e}(X_{\setminus x_i}) \leftarrow \sum_{v \in V_e} P(v|X_{\setminus x_i})$
    $\phi_i \leftarrow P_{V_e}(X) - P_{V_e}(X_{\setminus x_i})$
    $\mathcal{L}_{wm}^{(i)} \leftarrow \max(0, \epsilon - w_i \cdot \phi_i)$
    $g \leftarrow g + \nabla \mathcal{L}_{wm}^{(i)}$ ▷ Backpropagate and release this graph
  **end for**
  Update $f$ parameters using $g$
**end for**
**return** $f^* \leftarrow f$

**Algorithm 2** Watermark Extraction

**Require:** Suspicious LLM $\hat{f}$, trigger prompt $X = \{x_1, \ldots, x_k\}$, secret key $sk$, sampling times $T$, verification threshold $\tau$, ground truth watermark $w$
**Ensure:** Copyright Decision (Stolen or Independent)

$V_e \leftarrow$ recover vocabulary subset using $sk$
Query $\hat{f}$ with $X$ for $T$ times to obtain tokens $\{y_1, \ldots, y_T\}$
$\hat{P}_{V_e}(X) \leftarrow \frac{1}{T} \sum_{j=1}^{T} \mathbb{I}(y_j \in V_e)$ ▷ Frequency-based estimation
**for** $i = 1$ to $k$ **do**
  $X_{\setminus x_i} \leftarrow$ replace $x_i$ in $X$ with neutral token $t$
  Query $\hat{f}$ with $X_{\setminus x_i}$ for $T$ times to obtain $\{y'_1, \ldots, y'_T\}$
  $\hat{P}_{V_e}(X_{\setminus x_i}) \leftarrow \frac{1}{T} \sum_{j=1}^{T} \mathbb{I}(y'_j \in V_e)$
  $\hat{\phi}_i \leftarrow \hat{P}_{V_e}(X) - \hat{P}_{V_e}(X_{\setminus x_i})$
  $\hat{w}_i \leftarrow \text{sign}(\hat{\phi}_i)$
**end for**
WSR $\leftarrow \frac{1}{k} \sum_{i=1}^{k} \mathbb{I}(\hat{w}_i = w_i)$
**if** WSR $\geq \tau$ **then**
  **return Stolen** (Copyright Infringement)
**else**
  **return Independent**
**end if**

**Example Trigger Sample**

we 've been spending a lot of time in los angeles talking to tv production people says mike parks president of call interactive which supplied technology for both abc sports and nbc 's consumer minutes with the competitiveness of the television market these days everyone is looking for a way to get viewers more excited ...

Figure 3: Example trigger sample in Ptb-text-only.

Table 7: Experimental results of CAST across additional LLMs. The "Fidelity" metric is evaluated on MMLU. In the "Robustness" evaluation, we report the WSRs in the text-only scenario after fine-tuning.

| Model | Effectiveness | | | Fidelity | | Robustness | |
|---|---|---|---|---|---|---|---|
| | Full Logits | Top-10 | Text-only | No WM | WM | Alpaca | Dolly |
| Llama-3.2-1B-Instruct | 100.0 | 100.0 | 100.0 | 46.00 | 46.49 (+0.49) | 96.88 | 90.62 |
| Qwen2.5-1.5B-Instruct | 100.0 | 100.0 | 100.0 | 60.17 | 60.22 (+0.05) | 90.62 | 87.50 |
| Qwen2.5-3B-Instruct | 100.0 | 100.0 | 100.0 | 65.49 | 65.27 (-0.22) | 90.62 | 84.38 |
| Llama-2-7B-Chat | 100.0 | 100.0 | 100.0 | 46.25 | 45.54 (-0.71) | 87.50 | 87.50 |
| Mistral-7B-Instruct-v0.3 | 100.0 | 100.0 | 100.0 | 59.71 | 59.25 (-0.46) | 93.75 | 87.50 |

# C Experiment Details

**Detailed Settings for** CAST**.** When implementing CAST, to maintain the utility, we randomly select 2000 samples from Ptb-text-only to calculate $\mathcal{L}_{\text{CE}}$. We also select one 128-token sample (an example in Figure 3) from Ptb-text-only as the trigger sample while embedding watermarks with less than 128 bits, and one 256-token sample when embedding a 256-bit watermark. The target watermark is randomly generated. In our experiments, we use the <unk> token or the 10,000-th token (if there is no <unk> token in the vocabulary) as the neutral replacement token $t$. The size of the target explaining vocabulary subset $V_e$ is half of the whole vocabulary $V$. We fine-tune the LLMs for 20 epochs. The balancing coefficient $\lambda$ is set to $1.0$, and the default $\epsilon$ is set to $0.20$. In all the experiments, we utilize LoRA (Hu et al., 2022) to fine-tune the model, with the rank $r = 8$, the scaling factor $\alpha = 16$, and the dropout rate $= 0.1$.

**Detailed Settings for the Baseline Methods.** In our experiments, we take two output-space watermarking methods and two alternative-space watermarking methods as our baselines. The hyperparameter settings of these methods are the same as those used in their original papers. For EverTracer, we utilize the AG News (Zhang et al., 2015) dataset to fine-tune the model to enhance memorization. For EaaW, since the output logits of input tokens cannot be obtained in the black-box, API-level auditing scenario, we modify the original approach, "computing the average logits of all target tokens in the sequence", to "computing the logits of the target token while generating the first new token."

**Software and Hardware Settings.** All the code is developed with Python 3.11 and PyTorch. We utilize at most 4 NVIDIA RTX 4090 48G GPUs in our experiments.

# D Additional Experiments

## D.1 Experiments on Additional LLMs

In this section, we evaluate the generalizability of CAST by extending our experiments to a broader range of modern LLMs with varying parameter sizes and architectures. Specifically, we include five additional models: Llama-3.2-1B-Instruct (Grattafiori et al., 2024), Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct (Team, 2024), Llama-2-7B-Chat (Touvron et al., 2023), and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023). We maintain the same experimental settings as Section 4.

As presented in Table 7, CAST demonstrates exceptional scalability and stability across all metrics. In terms of effectiveness, the method achieves a perfect 100.0% WSR across all auditing scenarios (Full Logits, Top-10, and Text-only) for every model tested. Regarding fidelity, the watermarking process preserves model utility, with MMLU score fluctuations remaining negligible (ranging from $-0.71$ to $+0.49$) compared to unwatermarked baselines. Furthermore, CAST exhibits strong robustness against fine-tuning attacks (Alpaca and Dolly), maintaining practical extractability (WSR $> 84\%$) even in the strict text-only setting. These results confirm that CAST provides a robust and utility-preserving solution across diverse open-source LLMs.

Table 8: Additional fidelity evaluation on generative benchmarks and MMLU token distributions. JS divergences are medians over the evaluation samples.

| Method | XSum | GSM8K | HumanEval | First-token JS | All-token JS |
|---|---|---|---|---|---|
| No Watermark | 0.128 | 0.854 | 0.750 | / | / |
| CAST | 0.128 | 0.855 | 0.732 | $1.9\times10^{-9}$ | $4.4\times10^{-10}$ |

Table 9: WSR (%) under prolonged fine-tuning and model merging. The merge ratio denotes the fraction of the watermarked model.

| Attack | Scenario | Attack strength | | | | |
|---|---|---|---|---|---|---|
| Fine-tuning | Steps | 0 | 80 | 160 | 320 | 640 |
| | Full Logits | 100 | 100 | 98 | 98 | 98 |
| | Top-10 | 100 | 100 | 98 | 98 | 98 |
| | Text-only | 98 | 92 | 94 | 92 | 94 |
| Model merging | Merge ratio | 0.3 | 0.5 | 0.7 | | |
| | Full Logits | 80 | 88 | 99 | | |
| | Top-10 | 81 | 89 | 97 | | |
| | Text-only | 80 | 87 | 98 | | |

### D.2 Additional Fidelity Evaluations

The main evaluation focuses on multiple-choice benchmarks. To test whether watermark embedding affects open-ended generation or the output distribution, we additionally evaluate summarization on XSum (Narayan et al., 2018), mathematical reasoning on GSM8K (Cobbe et al., 2021), code generation on HumanEval (Chen et al., 2021), and the Jensen–Shannon (JS) divergence between the original and watermarked token distributions on MMLU. Table 8 shows that CAST preserves the scores of the original model on XSum and GSM8K, with only a small fluctuation on HumanEval. The median JS divergence is below $10^{-8}$ for both the first generated token and all generated tokens, indicating that the watermark causes little distributional shift on normal inputs.

### D.3 Additional Robustness Evaluations

**Longer Fine-tuning and Model Merging.** Table 9 extends downstream fine-tuning to 640 steps and evaluates merging the watermarked model with its unwatermarked counterpart. The watermark remains stable under prolonged fine-tuning. Under model merging, detectability increases with the fraction of the watermarked model, and remains above the random-matching baseline in all evaluated settings.

**Prefix Manipulation.** We also test a provider that forces every response to begin with a fixed prefix. Table 10 shows that this manipulation substantially weakens the watermark, but simultaneously degrades MMLU accuracy by more than 12 points. Thus, prefix manipulation exposes a watermark-removal–utility trade-off and is a limitation under stronger verification-time attacks.

### D.4 Ablation Study

#### D.4.1 Effect of Trigger Data

We evaluate the influence of different trigger datasets on the performance of CAST. We use Ptb-text-only (Marcus et al., 1993) as the default source and additionally sample triggers from Wikitext (Merity et al., 2017), Lambada (Paperno et al., 2016), and Tinystories (Eldan & Li, 2023). As shown in Table 11, CAST achieves near-perfect WSRs ($\geq 99.2\%$) across all trigger sources. The average fidelity scores also remain stable, ranging from 67.82 to 68.73, indicating that the method is insensitive to the choice of trigger data.

#### D.4.2 Effect of $\epsilon$

Table 10: Watermark effectiveness and model utility under prefix attacks.

| Metric | No Attack | Short Prefix | Long Prefix |
|---|---|---|---|
| WSR (%) | 100 | 59 | 55 |
| MMLU (%) | 69.74 | 56.80 | 56.16 |

Table 11: The effectiveness and fidelity of CAST using trigger prompts from different datasets.

| Trigger Data | #bits | Effectiveness | | | Fidelity |
|---|---|---|---|---|---|
| | | Full Logits | Top-10 | Text-only | Avg. |
| Ptb-text-only | 32 | 100.0 | 100.0 | 99.2 | 68.45 |
| | 64 | 100.0 | 100.0 | 99.2 | 68.67 |
| | 128 | 100.0 | 100.0 | 99.2 | 68.24 |
| Wikitext | 32 | 100.0 | 100.0 | 100.0 | 68.63 |
| | 64 | 100.0 | 100.0 | 100.0 | 68.05 |
| | 128 | 100.0 | 100.0 | 100.0 | 68.10 |
| Lambada | 32 | 100.0 | 100.0 | 100.0 | 68.73 |
| | 64 | 100.0 | 100.0 | 100.0 | 68.52 |
| | 128 | 100.0 | 100.0 | 99.2 | 67.82 |
| Tinystories | 32 | 100.0 | 100.0 | 100.0 | 68.68 |
| | 64 | 100.0 | 100.0 | 100.0 | 68.29 |
| | 128 | 100.0 | 100.0 | 100.0 | 68.08 |

We investigate the margin coefficient $\epsilon$ from 0.05 to 0.30. As shown in Figure 4, setting $\epsilon$ too low (*e.g.*, 0.05) yields insufficient embedding strength, particularly in the text-only scenario. Increasing $\epsilon$ enables reliable extraction while causing only minor utility fluctuations. These results show that CAST remains stable in fidelity across the evaluated margins while requiring sufficient embedding strength for text-only auditing.

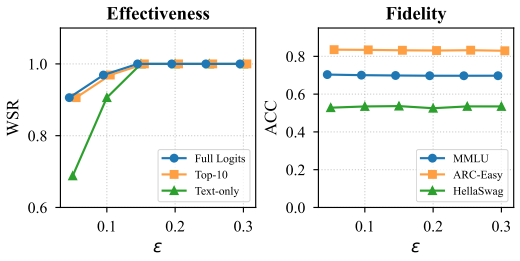


Figure 4: The effectiveness and fidelity of CAST with different $\epsilon$.

### D.5 Comparison with Recent Baselines

We compare CAST with iSeal (Xiong et al., 2026), which targets the stronger setting where verification evidence may be exposed to a malicious provider, and Semantically Conditioned Watermarks (SCW) (Gloaguen et al., 2026), which diffuse a statistical watermark throughout responses from a semantic domain. All methods are reproduced on Qwen3-4B-Instruct-2507. As shown in Table 12, CAST retains strong verification while preserving MMLU performance. The iSeal results should be interpreted together with its stronger threat model, whereas the SCW comparison shows that CAST induces a smaller token-distribution shift under this setup.

### D.6 Experiments on the Overhead of CAST

In this section, we measure the wall-clock time and peak GPU memory consumption for embedding watermarks of varying lengths into TinyLlama using EaaW and CAST. We use two GPUs with a combined memory of 96 GB. The results are in Table 13.

Regarding space efficiency, CAST significantly outperforms EaaW. CAST achieves $O(1)$ space complexity, and its memory consumption remains constant regardless of the watermark length. In contrast, EaaW scales linearly and quickly encounters out-of-memory (OOM) errors. Regarding time efficiency, both methods are highly practical. Although EaaW is slightly faster, the total overhead for both methods is below 35 minutes over 20 epochs.

Table 12: Comparison with recent baselines on Qwen3-4B-Instruct-2507. Token JS is the median Jensen–Shannon divergence on MMLU; "–" denotes an unavailable measurement.

| Method | WSR/WSR@50 (%) | MMLU (%) | Token JS |
|---|---|---|---|
| iSeal | 99 | 57.8 | – |
| iSeal (frozen LM head) | 73 | 66.7 | – |
| SCW | 100 | 69.76 | $5.3\times10^{-5}$ |
| CAST | 100 | 69.74 | $4.4\times10^{-10}$ |

Table 13: Time (min/epoch) and peak space (GB) overhead of EaaW and CAST.

| Method | Metric | 32-bit | 64-bit | 128-bit |
|---|---|---|---|---|
| EaaW | Time | 0.59 | 0.74 | / |
| | Space | 48.03 | 65.21 | OOM |
| CAST | Time | 0.78 | 1.10 | 1.73 |
| | Space | 37.46 | 37.46 | 37.46 |

# E Connections and Differences with Other Related Domains

In this section, we discuss the connections and differences between model watermarking and some related domains, such as AIGC watermarking and model fingerprinting.

## E.1 Comparison with AIGC Watermarking

AIGC watermarking focuses on embedding verifiable signals directly into the content generated by artificial intelligence models, ensuring the provenance and integrity of the output (Ren et al., 2024; Yang et al., 2025b). In the context of LLMs, these techniques are primarily categorized into training-free and training-based approaches (Yang et al., 2025b). Training-free methods, such as logits-based (Kirchenbauer et al., 2023) and score-based algorithms (Dathathri et al., 2024), operate during the inference stage by subtly adjusting token probabilities or sampling strategies to embed statistical signals without altering model parameters. Conversely, training-based methods involve modifying the model itself through unsupervised fine-tuning or the use of external decoders to ensure that the generated text intrinsically carries the watermark (Zhang et al., 2024c; Pang et al., 2025). These methods aim to balance robustness, imperceptibility, and the capacity to mitigate unauthorized usage of AIGC.

In contrast, model watermarking diverges significantly in its objectives and implementation. While AIGC watermarking targets the generated content to address issues like misinformation and plagiarism, model watermarking is designed to protect the intellectual property of the model artifact itself. This method typically embeds watermarks into the model's parameters or weights during the training process (Zhang & Koushanfar, 2024). The verification of a model watermark generally relies on specific interactions with the model, such as querying it with trigger inputs to elicit a pre-defined, distinct behavior (Xu et al., 2024), rather than analyzing the distribution of generated text in general scenarios. Consequently, the threat models differ: model watermarking defends against model theft and unauthorized deployment, whereas AIGC watermarking serves as a safeguard for content attribution and the traceability of model-generated outputs.

## E.2 Comparison with Non-intrusive LLM Fingerprinting

LLM fingerprinting serves as a post-hoc auditing mechanism that identifies whether a suspicious model is derived from a source model by extracting inherent, distinctive characteristics rather than embedding external signals. In black-box scenarios, these techniques are categorized into two distinct paradigms: untargeted and targeted fingerprinting (Shao et al., 2025a). Untargeted fingerprinting (Pasquini et al., 2025; Gao et al., 2025; McGovern et al., 2025) evaluates the behavioral similarity of models by analyzing features of their responses (such as probability distributions and text embeddings) to general query sets, without relying on predefined triggers. Conversely, targeted

fingerprinting (Gubri et al., 2024; Jin et al., 2024; Tsai et al., 2025) relies on identifying specific input-output pairs. It typically utilizes adversarial optimization to construct specific "query-response pairs" that elicit unique, pre-determined outputs ($e.g.$, a specific error or number) exclusively from the source model.

The primary advantage of fingerprinting is its non-intrusive nature. Unlike LLM watermarking, which modifies parameters and can degrade model performance, fingerprinting preserves the model's integrity and is applicable to already released models. However, despite these advantages, recent comprehensive evaluations (Shao et al., 2025a) reveal that current black-box fingerprinting methods remain unreliable for practical deployment. Empirical results indicate that these methods struggle with robustness, exhibiting low effectiveness and high false positive rates when facing standard modifications like fine-tuning. Because existing fingerprinting techniques often fail to provide the high-confidence detection required for real-world auditing, this work primarily focuses on LLM watermarking technology as a more stable solution for LLM copyright protection.

## F Limitations and Future Direction

Despite the promising results of CAST, there remain some avenues for future improvement. First, while CAST removes the watermark-length-dependent peak auxiliary-memory bottleneck of prior explanation-based methods, token-wise attribution still requires computation that grows with the watermark length. Future work could accelerate embedding without compromising precision. Second, prefix manipulation reveals that a provider willing to substantially degrade model utility can weaken text-only verification. More fundamentally, no text-only black-box method can identify the deployed model if a malicious API perfectly recognizes every verification request and fabricates arbitrary responses; public verification in this stronger setting requires complementary trusted mechanisms. Finally, this work primarily targets text-modal LLMs. Extending the mechanism to multimodal models ($e.g.$, Vision-Language Models) represents an exciting opportunity for broader application.

## G Impact Statement

In terms of positive societal impact, this paper aims to address the critical challenges in copyright protection for LLMs by introducing a scalable and utility-preserving watermarking framework, CAST. CAST could assist both academia and industry in safeguarding the intellectual property rights of high-value models against unauthorized replication and theft. Furthermore, by establishing a trustworthy technical foundation for copyright auditing, this work contributes to a sustainable AI ecosystem, potentially facilitating the emergence of legitimate model sharing and commercialization mechanisms. Regarding the negative impact, our research is motivated by defensive objectives. As a passive auditing tool designed solely for provenance verification, we believe there are no significant or notable negative societal impacts associated with this work.