# Resilient Output-Space Watermarking for Language Models via Null-Space Invariants

**Anonymous ACL submission**


## Abstract

Language models (LMs) have emerged as high-value intellectual property (IP) assets, necessitating robust copyright protection. Given that many watermarking methods embed information into the last-layer hidden states, this paper analyzes and extends Linear Functionality Equivalence Attacks (LFEA) to last-layer hidden states, a new attack we call LL-LFEA. We identify a critical security gap: existing output-based watermarking schemes are also vulnerable to LL-LFEA, with their verification metric become invalid after being significantly altered, while preserving downstream accuracy (within 2% of the original model). To counter this, we discover that the *null space* of the LM's output matrix remains invariant under any LL-LFEA transformation. Leveraging this insight, we propose NSmark, a task-agnostic watermarking framework that embeds ownership information simultaneously in the last-layer outputs and their null space. NSmarkconsists of three phases: (i) watermark generation using a digitally signed identifier with spread-spectrum modulation; (ii) watermark embedding through a lightweight output-mapping extractor that preserves model fidelity; and (iii) watermark verification using both extraction rate and a null-space conformity metric. Extensive experiments demonstrate that NSmarkremains verifiable under LL-LFEA, and incurs negligible computational overhead. Code is available at https://anonymous.4open.science/r/NSmark-2FC1/README.md.


## 1 Introduction

Over the past few decades, language models (LMs) have achieved exceptional performance and found applications across a wide range of fields. However, training high-performance LMs incurs prohibitive costs in terms of massive datasets, computational power, and expert knowledge. Consequently, these models have become high-value intellectual property (IP) assets. As the commercialization of LMs accelerates, often via API services or direct parameter licensing, the risk of unauthorized distribution, theft, and misuse has increased. This creates an urgent demand for robust digital rights management mechanisms to verify model ownership and trace copyright infringements.

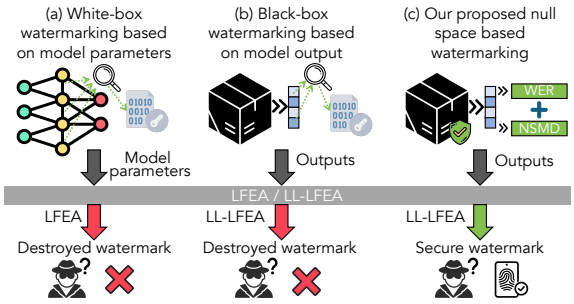


Figure 1: Illustration of different watermark schemes against LFEA/LL-LFEA. LFEA disables parameters based white-box schemes and LL-LFEA disables output based black-box schemes (in Section 3). NSmarkis secure against LL-LFEA using null space invariance.

Watermarking techniques have been widely used to protect the IP of models (Chen et al., 2024; He et al., 2024; Carlini et al., 2024; Feng et al., 2024) and model-generated contents (Kirchenbauer et al., 2023; Zhao et al., 2023; Qu et al., 2025). In this work, we focus on the IP protection of the model itself. By incorporating identifiable information, these techniques could verify model ownership and provide proof of authenticity. However, protecting the IP of LMs through watermarking presents significant challenges. Since LMs can be deployed for various post-training downstream tasks (Zhang et al., 2023), for example, Embeddings-as-a-Service (EaaS), which underpins many applications; therefore, it is essential that watermarking schemes remain task-independent.

Critically, we identify a fundamental vulnerability in current protection mechanisms. While the Linear Functionality Equivalence Attack (LFEA) (Li et al., 2023a) was originally designed for white-box parameters, we demonstrate that

a similar principle applies to black-box outputs. Since the last hidden states (hereafter "output") in LMs are widely used in classification (Kenton and Toutanova, 2019), sentence embeddings (Shetty et al., 2024b), and generation (Touvron et al., 2023), we consider a scenario (illustrated in Figure 1) where a model produces a watermarked output vector. An attacker can inject a linear transformation matrix to create a perturbed vector. By applying the inverse transformation in the subsequent downstream layer, the attacker ensures the final prediction remains identical, preserving utility. However, this transformation scrambles the spatial distribution of the watermark signal, we call this Last-Layer LFEA (dubbed LL-LFEA). We show that attackers can manipulate output vectors to destroy embedded watermarks without degrading downstream performance, revealing a significant blind spot in current defense strategies.

Our preliminary experiments show that under LL-LFEA, the verification metrics of recent methods like EmbMarker (Peng et al., 2023) and WARDEN (Shetty et al., 2024a) are changed rendering the watermark undetectable. To defend against this, we first explore the characteristics of the outputs. Our key observation is that while linear transformations change the position of feature vectors, the null space of the output matrix exhibits strictly invariant properties under valid linear functionality equivalence. Based on this theoretical finding, we propose a new null space verification method that can withstand the LL-LFEA attack. This method uses a new metric, the Null Space Matching Degree (NSMD). NSMD measures the degree of match between the output matrix of the suspicious model and the null space of the original LM. Finally, we propose NSmark, a null-space-based task-agnostic watermarking scheme for protecting LMs. Unlike traditional methods that only embed watermarks directly into sensitive feature dimensions, NSmark-further utilizes the orthogonal null space for verification. Specifically, NSmarkuses identity information to generate all elements related to the watermark and uses the Watermark Extracting Rate (WER) and NSMD to verify the watermark, so it can work during model transactions and auditing (Li et al., 2022) as shown in Figure 1. Spread spectrum modulation technology and an extra extractor are also introduced to enhance watermark performance. Our contributions are summarized as follows:

- We identify the vulnerability of output-based watermarks and formalize the LL-LFEA attack. We prove that LL-LFEA can effectively invalidate existing watermarks in the embedding space while maintaining downstream utility.
- We derive the mathematical proof that the null space of the model's output matrix remains invariant under LL-LFEA. Based on this we propose a new watermark method NSmark, which is task-agnostic and uses both new null space verification and signature verification to resist LL-LFEA.
- We conduct comprehensive experiments by applying NSmarkto various models of pre-training and downstream tasks. The experimental results demonstrate the effectiveness, fidelity, reliability, and robustness of NSmark.

## 2 Related Work

### 2.1 Watermarking for LMs.

With the rise of pre-training in NLP, recent work has explored watermarking specific to LMs. BadPre (Jia et al., 2022) introduced a task-agnostic backdoor attack only for MLM-based LMs. Hufu (Xu et al., 2024) introduced a modality-agnostic approach for pre-trained Transformer models using the permutation equivariance property. Explanation as a Watermark (Shao et al., 2024) addressed the limitations of backdoor-based techniques by embedding multi-bit watermarks into feature attributions using explainable AI. (Shetty et al., 2024b; Peng et al., 2023; Shetty et al., 2024a) proposed EaaS watermarks to protect the intellectual property of EaaS providers. (Zhang et al., 2023; Shen et al., 2021) proposed task-agnostic backdoor attacks by assigning high-dimensional vectors as trigger set labels, but their effectiveness is sensitive to downstream classifier initialization. (Wang and Kerschbaum, 2021) introduced an auxiliary neural network for watermark embedding using weights from the main network. (Wu et al., 2022) proposed a task-agnostic embedding loss function, but did not consider the need for triggers to reflect the model owner's identity. (Cong et al., 2022) introduced a black-box watermarking scheme for PLMs, but its applicability is limited due to the discrete nature of word tokens. Unfortunately, these schemes are, in principle, vulnerable to attacks such as LFEA. In addition, (Chen et al., 2025) proposed the set of harmless perturbations in input and suggested that it can be used for the model fingerprint. Unfortunately, these schemes are, in principle, vulnerable to linear transforma-

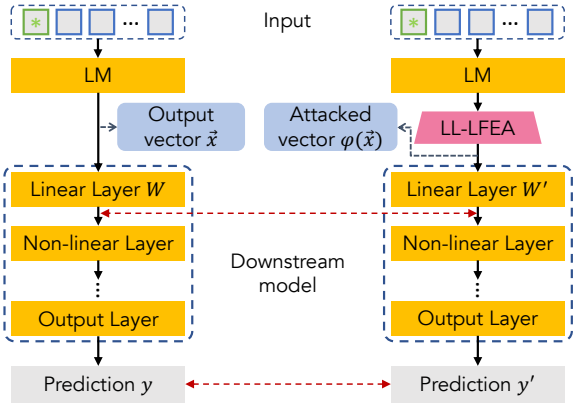


Figure 2: The diagram of model inference flow w/ and w/o LL-LFEA attack. LL-LFEA transforms the LM output and adds an inverse transform in the subsequent linear layer, leaving the final prediction unchanged.

tion attacks. Unlike these approaches, NSmarkdoes not rely solely on the feature in output vectors. By anchoring the verification in the orthogonal null space, our method remains robust even when the feature space is linearly shifted.

### 2.2 Watermark Removal Attacks.

DNN watermarking faces various removal attempts. Common methods include fine-tuning (Adi et al., 2018) and pruning (Han et al., 2015). Fine-pruning (Liu et al., 2018) combines these approaches for greater effectiveness. Knowledge distillation techniques (Hinton, 2015) can also inadvertently remove watermarks while reducing model size. (Shetty et al., 2024b) show that existing EaaS watermarks can be removed by paraphrasing attack. (Lukas et al., 2022) propose a new attack method called Neuron Reordering to swap neurons within the same hidden layer of a DNN to disrupt embedded watermarks in model parameters. (Li et al., 2023a) introduce a powerful LFEA for white-box watermarks, applying linear transformations to model parameters, effectively destroying embedded watermarks while preserving the model's original functionality. While LFEA focuses on inner model parameters, we extend this threat model to the output space (LL-LFEA), making LFEA so effective against traditional watermarks. Fraud attacks include overwriting (Wang and Kerschbaum, 2019) and ambiguity attacks (Zhu et al., 2020) also pose a great threat to watermarks.

## 3 Threat: LL-LFEA Attack Paradigm

Model watermarking has emerged as a promising solution for IP protection, embedding identifiable signals into model parameters (white-box) or outputs (black-box). In white-box watermarking, high-dimensional model parameters are often embedded with watermark information. For transformer-based LMs, the last hidden states (output) are also high-dimensional, so they can be watermarked similarly to white-box schemes (Peng et al., 2023; Shetty et al., 2024a; Li et al., 2023b, 2021). Despite their popularity, recent studies have exposed severe vulnerabilities in these schemes. Specifically, the LFEA (Li et al., 2023a) demonstrates that attackers can linearly transform intermediate model parameters to erase watermarks while preserving the model's functionality. Given that the last hidden states (outputs) are linear representations used for downstream tasks, we analyze and extend the LFEA paradigm to these outputs, formulating the Last-Layer LFEA. *Next, we show that attackers can manipulate output vectors to destroy embedded watermarks without degrading downstream performance, revealing a significant blind spot in current defense strategies.*

Assume that the attacker knows that the watermark is embedded in the output and seeks to remove the watermark with minimal cost while ensuring that normal performance remains unaffected. As shown in Figure 2, the output vector $\vec{x}$ is generated by the last layer of the LM transformer and serves as input to the downstream modules. For LMs such as BERT (Kenton and Toutanova, 2019), $\vec{x}$ tend to be fed into the *Output Linear Layer (LM Head)* to predict token logits, or used as a feature representation for input to tasks such as classification. For the former case, since the subsequent layers are part of the LM itself and $\vec{x}$ is an intermediate output, it can be viewed as a white-box case covered by the LFEA attack. For the latter case, $\vec{x}$ is the final output of the LM, and the subsequent layers do not belong to the LM itself, which can be viewed as a black-box case beyond the scope of LFEA, so we analyze and expand the risks as follows. After passing $\vec{x}$ through a series of linear and non-linear layers, the prediction result $y$ is obtained. The attacker attempts to modify $\vec{x}$ to destroy the watermark while ensuring that the final prediction remains unaffected. Specifically, the attacker changes $\vec{x}$ to $\varphi(x)$ and inputs it into the modified downstream model, so that the resulting prediction $y'$ remains equal to the original prediction $y$.

The sufficient condition for this is that the change to $\vec{x}$ is compensated for after passing through the first linear layer of the downstream net-

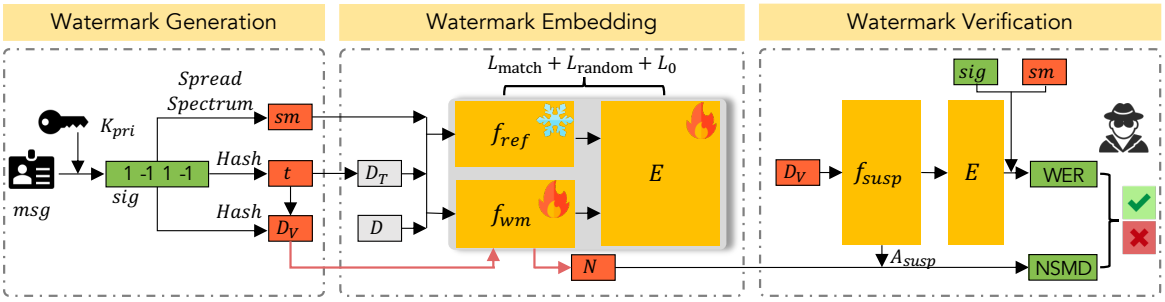


Figure 3: The workflow of NSmark. (i) In watermark generation, identity information is used generate $sig$. (ii) In watermark embedding, watermarked model $f_{wm}$ and extractor $E$ are trained with the participation of the reference model $f_{ref}$. (iii) In watermark verification, WER and NSMD collaborate to verify the identity of the model.

work (denoted as $W$). In this case, the attacker aims to satisfy the following condition: $W'\varphi(\vec{x}) = W\vec{x}$, which leads to $\varphi(\vec{x}) = W'^{,\dagger}W\vec{x} = Q\vec{x}$, where $Q = W'^{,\dagger}W$ and $W'^{,\dagger}$ is the pseudo-inverse of $W'$ (Li et al., 2023a). To avoid loss of information during the linear transformation (since this would adversely affect downstream tasks), $Q$ must be a reversible matrix. We present a simple method and analysis on how to quickly generate high-dimensional $Q$ in Appendix A.

*We show that the attacker can apply a linear transformation to $\vec{x}$ thereby destroying the watermark, while leaving the downstream task performance unchanged. We refer to this attack as the Last-Layer LFEA (LL-LFEA).*

## 4 Watermark Method

### 4.1 Null Space Verification Theory

LL-LFEA applies a linear transformation to the LM output vector and can destroy the watermark. As a result, previous watermark verification methods may be significantly affected. From the perspective of model input, (Chen et al., 2025) studies harmless perturbations. We observe from the output perspective that the null space of the matrix composed of the output vector is invariant under the LL-LFEA attack. Based on this, we propose using the null space matching degree to verify whether the model is embedded with watermarks.

**Theorem 1.** *Let $A_1 \in \mathbb{R}^{n\times m}$ be the matrix of the watermarked LM outputs on the trigger dataset (with $N_1$ its null space matrix) and let $A_2 = Q \times A_1$ be the corresponding output matrix after applying LL-LFEA (with $Q$ invertible). Then $N_1$ is the null space of $A_2$. (Proofs in Appendix B.)*

Therefore, even if the watermark based on the digital string is corrupted, we can still verify model ownership using the null space of the output matrix.

### 4.2 Null Space Match Degree (NSMD)

We define NSMD by introducing a matrix, which is obtained by multiplication of the matrix of the output matrix $A$ of any LM without watermark and the null space matrix $N$ of $f_{wm}$, where $f_{wm}$ means watermarked model. We first normalize all row vectors $\alpha_i, i \in [1, n]$ of $A$ and all column vectors $\beta_j, j \in [1, p]$ of $N$ so that $\alpha$ and $\beta$ are distributed on the unit sphere. In $H_{(n\times p)} = A_{(n\times m)} \times N_{(m\times p)}$, $H_{i,j} = \alpha_i \cdot \beta_j$ is the dot product of the $i$th row vector of $A$ and the $j$th column vector of $N$. We define NSMD of $A$ and $N$ as:

$$\text{NSMD}(A, N) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}\sqrt{|H_{i,j}|}. \quad (1)$$

**Next, we give a detailed analysis of estimation of NSMD in Appendix C.** For example, if $n = 768$ and $p = 1500$, we have NSMD $> 27.48$. This difference is amplified by the square root calculation process, resulting in a significant difference between whether $A$ and $N$ are matched. We use this difference to distinguish whether the model is embedded with a watermark.

### 4.3 Framework of NSmark

As shown in Figure 3, NSmarkincludes three phases: watermark generation, watermark embedding, and watermark verification.

#### 4.3.1 Watermark Generation

Algorithm 1 introduces watermark generation. Many mechanisms are used to add user information to watermarks. First, the digital signature $sig = \textbf{Sign}(msg)$ is generated from the identity information message $msg$. To ensure that the trigger $t$ has a unique mapping relationship with $sig$, only one trigger is used. We use the trigger generation algorithm **Encode**$(\cdot)$ introduced in Li et al. (2023b)

to obtain $t = \textbf{Encode}(sig, n = 1)$. $t$ is inserted into clean sample $x$ of dataset $D$ to form a trigger set $D_T$. To defend against ambiguous attacks, the verification trigger set $D_V$ used for null space verification should also be generated from $sig$. The detailed method is introduced in Appendix D.2 .

To improve the robustness of the watermark, we introduce the spread spectrum modulation technology as (Feng and Zhang, 2020). Spread spectrum modulation technology uses redundant bits to represent the original information. Figure 6 shows an example of the 3× spreading. Please refer to Appendix D.4 for the specific process $\textbf{SM}(sig) \rightarrow sig_{wm}$.

#### 4.3.2 Watermark Embedding

Before the training starts, make a copy of $f_{wm}$ as the frozen reference model $f_{ref}$. Then use the clean data set $D$ and the trigger set $D_T$ to train $f_{wm}$ and the extractor $E$, where $E$ is a linear network. When taking $\{D, D_T\}$ as input, $f_{wm}$ will output $\{V, V^T\}$ and $f_{ref}$ will output $\{V_{ref}, V_{ref}^T\}$, respectively. For $V^T$, $E$ maps it to obtain the signature $sig_{wm}$, and for $\{V, V_{ref}, V_{ref}^T\}$, $E$ maps them to random vectors. After the training is completed, the watermark is embedded in $f_{wm}$. Then using $D_V$ as input, the output vectors are concatenated into a matrix $A$, and the corresponding null space matrix $N$ of $A$ is calculated as part of the key.

Three networks are involved in watermark embedding: the model $f_{wm}$ to be embedded with the watermark, the reference model $f_{ref}$ and the extractor model $E$. Compared to directly embedding $sig$ in the output vector of $f_{wm}$, adding $E$ to the map can reduce the side effect of the watermark on the original performance. The watermark capacity is increased at the same time. We use the mean square error loss (MSE) and the similarity function sim to implement the above training process:

$$L_{\text{match}} = \frac{1}{|D_T|} \sum_{x \in D_T} \text{MSE}\left(E\left(V^T\right), sig_{sm}\right), \quad (2)$$

$$\begin{aligned} L_{\text{random}} &= \frac{1}{|D|} \sum_{x \in D} \text{sim}\left(E\left(V\right), sig_{sm}\right)^2 \\ &+ \frac{1}{|D_T|} \sum_{x \in D_T} \text{sim}\left(E\left(V_{ref}^T\right), sig_{sm}\right)^2 \\ &+ \frac{1}{|D|} \sum_{x \in D} \text{sim}\left(E\left(V_{ref}\right), sig_{sm}\right)^2. \end{aligned} \quad (3)$$

We use cosine similarity as the sim function. The complete loss function of $E$ is $L_{\text{Extractor}} = \lambda_1 L_{\text{match}} + (1 - \lambda_1) L_{\text{random}}$, where $L_{\text{random}}$ ensures that no $sig_{sm}$ is generated on the non-triggering set. During training, only the parameters of $E$ are trainable. The loss of $f_{wm}$ also consists of two parts: $L_{wm} = \lambda_2 L_{match} + (1 - \lambda_2) L_0$. The content of this $L_{match}$ is the same as $L_{match}$ of $E$, but only the parameters of $f_{wm}$ are updated at this time, and $L_0$ is the original LM training loss function. During training, $E$ and $f_{wm}$ are trained alternately.

#### 4.3.3 Watermark Verification

To effectively defend attacks, NSmarkuses two metrics together to verify ownership: WER and NSMD. Model owner submits $key = (sig, E, N)$ to the Certification Authority (CA). CA generates $t, D_V, sm$ using $sig$. Input $D_V$ into the suspicious model $f_{susp}$ to get the output vector $A_{susp}$, and pass $A_{susp}$ through $E$ to get the mapped vector $O_{susp}$. Then WER is obtained from the despread spectrum in Appendix D.5.

WER is defined from comparing each bit $a_i$ and $a_i'$ in $sig$ and $sig'$:

$$\text{WER} = \frac{1}{n} \sum_{i=0}^{n-1} \left[a_i = a_i'\right], \quad (4)$$

where $[\cdot]$ is the *indicator function*, which is 1 when the expression in the bracket is *True*, otherwise 0.

NSMD is calculated from $A_{susp}$ and $N$ using Equation 1. We define two thresholds, and whether WER$> T_W$ will be first verified. If it fails, whether NSMD$< T_N$ will be further considered in the case of LL-LFEA.

## 5 Experiments

### 5.1 Experimental Setup

We evaluate NSMARK from three aspects: watermark verifiability, model fidelity, and robustness against watermark removal attacks. We use WikiText-2 (Merity et al., 2017) for watermark embedding, and evaluate downstream utility on five classification datasets: SST-2 and SST-5 (Socher et al., 2013), Lingspam (Sakkis et al., 2003), OffensEval (Zampieri et al., 2019), and AG News (Zhang et al., 2015). Unless otherwise stated, the main experiments are conducted on the base versions of BERT (Kenton and Toutanova, 2019), RoBERTa (Liu, 2019), DeBERTa (He et al., 2020), and XLNet (Yang, 2019); additional experiments on larger models are reported in Appendix N. We report downstream accuracy (ACC) for model utility, and use Watermark Extraction Rate (WER) and

Null Space Matching Degree (NSMD) for watermark verification. Detailed hyperparameters, extractor architecture, trigger construction, and training configurations are provided in Appendix E.

#### 5.1.1 Datasets

We use WikiText-2 (Merity et al., 2017) for watermark embedding. Among the applications of last hidden states, we select the most common classification tasks to evaluate the performance of watermarks on downstream tasks: SST-2 and SST-5 (Socher et al., 2013) for sentiment analysis, Lingspam (Sakkis et al., 2003) for spam detection, OffensEval (Zampieri et al., 2019) for offensive language identification and AG News (Zhang et al., 2015) for news classification.

### 5.2 LL-LFEA Attack Evaluation

We select the effective *embedding-based watermarking scheme (EmbMarker (Peng et al., 2023), WARDEN (Shetty et al., 2024a), WET (Shetty et al., 2024b))*, invariance-based watermarking (Hufu (Xu et al., 2024)) and NSmark(without NSMD) as victims to study the effectiveness of LL-LFEA.

The attack results are shown in Appendix F.

### 5.3 NSmarkPerformance Evaluation

#### 5.3.1 Effectiveness

Effectiveness means that the watermark can achieve the expected effect during verification. Ideally, $sig'$ extracted from the watermarked model should be consistent with the original $sig$, and the output matrix of $f_{wm}$ for $D_V$ should match completely $N$ stored in $key$, which means WER = 1 and NSMD = 0. Table 1 shows the results of $f_{wm}$ embedded with watermark and $f_{clean}$ without watermark. It can be seen that for different watermarked LMs, WER is 1 and NSMD is close to 0. This shows the effectiveness of NSmark. Comparison of the values of $f_{wm}$ and $f_{clean}$ shows that WER and NSMD will obviously change after the watermark is embedded. Although NSMD of different LMs has different values, they are all far from 0. Through these results, we can preliminarily define verification thresholds of WER and NSMD as $T_W = 0.6$ and $T_N = 43$, which are $0.6\times$ the average gaps. Thresholds can be further adjusted according to different models and task types.

#### 5.3.2 Reliability

We further test whether NSmarkcan be falsely verified with incorrect keys. Results in Appendix G show that verification fails when any key component is incorrect: a wrong signature prevents trigger-aligned extraction, a wrong extractor cannot recover the embedded signature, and an incorrect null-space matrix yields a large NSMD. These results confirm that NSMARK binds ownership to the complete key tuple $(sig, E, N)$ rather than to any single component.

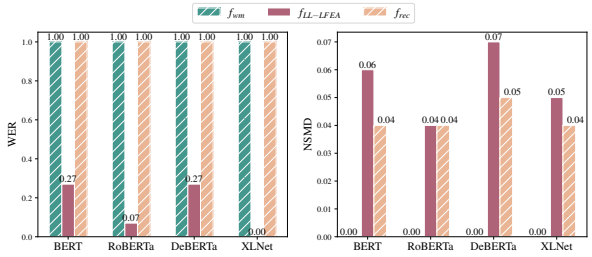


Figure 4: Changes of WER and NSMD before and after LL-LFEA attack and our proposed recovery. $f_{LL-LFEA}$ and $f_{rec}$ denote $f_{wm}$ attacked by LL-LFEA and recovered model from $f_{LL-LFEA}$, respectively.

#### 5.3.3 Fidelity

To test whether NSmarkaffect the performance on the original tasks, we add a downstream network to $f_{wm}$, and fine-tune the whole model $F_{wm}$ with the downstream dataset. $F_{clean}$ without watermark is fine-tuned as baseline. Table 2 shows that the watermark has almost no impact on the performance of the model on the original task.

#### 5.3.4 Defense against LL-LFEA

When designing NSmark, we focus on resisting LL-LFEA and propose null space verification using NSMD. Table 3 shows the impact of LL-LFEA on watermark verification, where $f_{LL-LFEA}$ denotes the model $f_{wm}$ attacked by LL-LFEA. Experiments show that after LL-LFEA, WER drops significantly, as discussed in Section 3, but NSMD is still close to 0, verifying that NSMD is an effective indicator for LL-LFEA. Furthermore, after applying LL-LFEA, the attacker can add a network to $f_{LL-LFEA}$ and fine-tune it for downstream tasks. (detailed results are presented in Appendix H)

Additionally, since LL-LFEA causes minimal degradation in model performance, the attacker may attempt to further compromise the watermark through multiple LL-LFEA attacks. Analysis for this aspect is provided in Appendix J.

• **Recovery of WER.** There is a method to recover the watermark after LFEA (Li et al., 2023a). We also propose a method to recover $f_{rec}$ from

| Metric | $f_{wm}$ | | | | $f_{clean}$ | | | |
|---|---|---|---|---|---|---|---|---|
| | **BERT** | **RoBERTa** | **DeBERTa** | **XLNet** | **BERT** | **RoBERTa** | **DeBERTa** | **XLNet** |
| WER | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.03 |
| NSMD | $2.94 \times 10^{-6}$ | $2.53 \times 10^{-6}$ | $2.91 \times 10^{-6}$ | $2.90 \times 10^{-6}$ | 60.95 | 61.24 | 87.88 | 76.74 |

Table 1: Effectiveness of NSmarkon different LMs. $f_{clean}$ means model w/o watermark. It shows that the signature can be accurately extracted (WER=1.00), and NSMD has obvious differentiation between $f_{wm}$ and $f_{clean}$.

| Metric | Model Setting | | SST-2 | SST-5 | Offenseval | Lingspam | AGnews |
|---|---|---|---|---|---|---|---|
| ACC | BERT | $F_{wm}$ | 91.40 | 52.62 | 85.12 | 99.14 | 93.95 |
| | | $F_{clean}$ | 91.63 | 53.03 | 84.07 | 99.66 | 94.38 |
| | RoBERTa | $F_{wm}$ | 92.55 | 54.71 | 84.30 | 99.66 | 94.43 |
| | | $F_{clean}$ | 94.04 | 56.15 | 84.88 | 100.00 | 94.72 |
| | DeBERTa | $F_{wm}$ | 93.00 | 55.48 | 83.02 | 99.31 | 94.61 |
| | | $F_{clean}$ | 93.58 | 57.65 | 85.12 | 99.31 | 94.84 |
| | XLNet | $F_{wm}$ | 88.65 | 42.67 | 81.98 | 99.14 | 93.29 |
| | | $F_{clean}$ | 93.58 | 53.62 | 84.65 | 99.31 | 94.07 |
| WER | BERT | $F_{wm}$ | 1.00 | 1.00 | 1.00 | 0.94 | 1.00 |
| | | $F_{clean}$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | RoBERTa | $F_{wm}$ | 0.98 | 1.00 | 1.00 | 0.72 | 0.99 |
| | | $F_{clean}$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | DeBERTa | $F_{wm}$ | 1.00 | 1.00 | 1.00 | 0.80 | 0.88 |
| | | $F_{clean}$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | XLNet | $F_{wm}$ | 1.00 | 1.00 | 1.00 | 0.92 | 1.00 |
| | | $F_{clean}$ | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| NSMD | BERT | $F_{wm}$ | 29.77 | 25.29 | 22.52 | 21.96 | 24.37 |
| | | $F_{clean}$ | 72.97 | 70.06 | 66.65 | 69.90 | 61.59 |
| | RoBERTa | $F_{wm}$ | 50.17 | 30.97 | 25.15 | 26.78 | 28.43 |
| | | $F_{clean}$ | 74.75 | 74.48 | 69.90 | 65.06 | 75.54 |
| | DeBERTa | $F_{wm}$ | 31.89 | 25.74 | 23.52 | 27.23 | 37.68 |
| | | $F_{clean}$ | 80.81 | 76.72 | 72.41 | 68.49 | 76.33 |
| | XLNet | $F_{wm}$ | 24.12 | 23.52 | 25.29 | 24.06 | 26.20 |
| | | $F_{clean}$ | 76.30 | 74.75 | 69.84 | 78.09 | 74.97 |

Table 2: Watermark evaluation against fine-tuning. $F_{wm}$ means fine-tuned whole watermarked model and $F_{clean}$ denotes fine-tuned model w/o watermark.

$f_{LL-LFEA}$. Specifically, assume that the output matrix of $f_{wm}$ is $A_{1(n\times m)}$ as Proof B. After being attacked with $Q_{(n\times n)}$, the output matrix turns to $A_2 = Q \times A_1$. Therefore, an estimate of $Q$ can be obtained as $Q' = A_2 \times A_1^{-1}$. If $m \neq n$, then $A_1$ is not reversible and $Q' = A_2 \times A_1^T \times (A_1 \times A_1^T)^{-1}$. Then we perform an anti-attack transformation on $f_{LL-LFEA}$, that is, multiply all the outputs of $f_{LL-LFEA}$ by $Q'$ to get $f_{rec}$. Figure 4 shows that after recovery, WER is significantly improved, indicating that such linear attacks are recoverable. In all cases, the NSMD is quite small, proving that NSMD is invariant to LL-LFEA. In the recovery algorithm of (Li et al., 2023a), both the attacker and the model owner might use such an algorithm to claim as the model owner, which will cause verification ambiguity. However, our proposed NSMD is invariant under LL-LFEA, so as long as the timestamp information is added to the $key$ tuple, the ownership can be reliably verified according to the time sequence of the model and the release of $key$.

### 5.3.5 Robustness

The robustness of watermark refers to whether watermark can be effectively verified after watermark removal attacks. Next, we will analyze the robustness of NSmarkagainst fine-tuning, pruning, fine-pruning, and overwriting attacks.

• **Robustness against fine-tuning.** Table 2 shows the results of WER and NSMD after fine-tuning on downstream tasks. $F_{wm}$ and $F_{clean}$ are obtained as in Section 5.3.3. In most cases, the WER is still very high, indicating that $sig$ can still be effectively extracted after fine-tuning. However, the WER of RoBERTa on Lingspam is relatively low. In main results we set the max input length to 128, which is quite shorter than the average length of Lingspam samples (695.26). Thus it is not sure the model's input includes triggers (possibly truncated). We perform further experiments on increasing the max length of input to 512, and modify the position of trigger to the front. WER increases to more than $0.84$ and $0.87$ respectively. Besides, compared to $F_{clean}$, there is still obvious discrimination. Therefore, for complex tasks, the verification threshold $T_W$ can be slightly lowered.

Figure 8 further illustrates the effect of different learning rates used in fine-tuning. The results indicate that learning rates have little impact on ACC and WER, maintaining watermark robustness. Similar to the results above, NSMD exhibits relatively more instability. However, the combined validation mechanism of the two metrics ensures that WER alone can still accurately validate the ownership of the model.

• **Robustness against pruning and fine-pruning.** Pruning is a commonly used model compression method and is often used to destroy the watermark embedded in the model. Referring to (Han et al., 2015; Shao et al., 2024), we sort the parameters of each layer in LM, then set some fractions of parameters with smaller absolute value to 0. Figure 5 shows that when the pruning rate is less than or equal to $0.8$, the WER is close to $1.0$. When

| Metric | BERT | | RoBERTa | | DeBERTa | | XLNet | |
|---|---|---|---|---|---|---|---|---|
| | $f_{wm}$ | $f_{LL-LFEA}$ | $f_{wm}$ | $f_{LL-LFEA}$ | $f_{wm}$ | $f_{LL-LFEA}$ | $f_{wm}$ | $f_{LL-LFEA}$ |
| WER | 1.00 | 0.27 | 1.00 | 0.07 | 1.00 | 0.27 | 1.00 | 0.00 |
| NSMD | $2.94 \times 10^{-6}$ | 0.06 | $2.53 \times 10^{-6}$ | 0.04 | $2.91 \times 10^{-6}$ | 0.07 | $2.90 \times 10^{-6}$ | 0.05 |

Table 3: LL-LFEA results on NSmark. $f_{LL-LFEA}$ denotes $f_{wm}$ attacked by LL-LFEA. Due to the null space invariance to linear transformation, NSMD can still effectively prove the IP of the model despite the failure of WER.

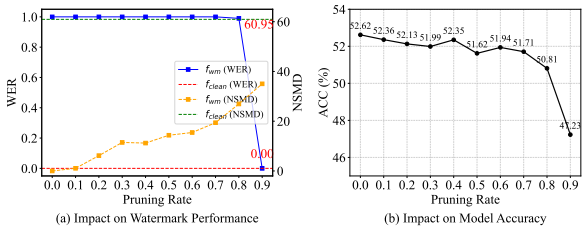


Figure 5: Evaluation of pruning attacks on NSmarkand main task performance.

| Model | Downstream Dataset | ACC | WER | NSMD |
|---|---|---|---|---|
| $f_{ow}$ | – | – | 1.00 | 22.76 |
| $F_{ow}$ | SST-2 | 92.32 | 0.98 | 48.77 |
| | SST-5 | 50.54 | 1.00 | 36.47 |
| | Offenseval | 84.77 | 1.00 | 36.61 |
| | Lingspam | 99.31 | 0.62 | 28.87 |
| | AGnews | 93.51 | 1.00 | 32.49 |

Table 4: Impact of overwriting attacks on watermark performance. $f_{ow}$ is the overwritten model, which is fine-tuned to obtain $F_{ow}$. "–" means not applicable.

the pruning rate is less than or equal to 0.6, NSMD does not change significantly, and even when the pruning rate is as high as 0.9, NSMD is still distinguishable. Besides, as shown in Appendix K, the accuracy of the watermarked model only changes slightly after pruning then fine-tuning on the SST-5. This shows that the embedded watermark is robust to pruning attack. Usually, pruning will affect the performance of the model on the original task, and the original task accuracy will be restored by fine-tuning (fine-pruning), as demonstrated by results in Appendix L.

• **Robustness against overwriting.** Overwriting means attacker embeds his own watermark into a model that has already been watermarked in the same way. This may destroy the original watermark. We simulate this process to obtain $f_{ow}$, then add a downstream network and fine-tune to obtain $F_{ow}$. We test the original watermark as shown in Table 4. The overwriting attack has little effect on ACC and WER except on Lingspam. Meanwhile, it has an impact on NSMD similar to that of fine-tuning.

• **Robustness against paraphrasing.** As proposed in (Shetty et al., 2024b), paraphrasing attack can bypass many watermark schemes. Thus, we study the robustness of proposed NSmarkagainst paraphrasing attack. In principle, paraphrasing attack changes the input text and the hidden state of the LM output, which may affect the extraction of WER. However, this does not change the semantic space to which the output matrix belongs, so the output space used to calculate NSMD will not change significantly. Referring to the original paper, we use DIPPER (Krishna et al., 2024) to paraphrase $P = 3$ on first 5000 lines of WikiText-2 and experiment on them. First watermarked model generates embedding, and the averaged embedding of multiple paraphrases are used to train the surrogate model. Results in Table 5 confirms our analysis, and our NSMD indicator is still below the threshold and could be used to verify the watermark.

| Model | BERT | RoBERTa | DeBERTa | XLNet |
|---|---|---|---|---|
| WER | 0.00 | 0.00 | 0.00 | 0.00 |
| NSMD | 15.56 | 15.11 | 14.67 | 13.13 |

Table 5: Results of robustness against paraphrasing attack.

## 6 Conclusion

This paper proposes NSmark, a backdoor based watermarking framework to verify the ownership of LMs. We first analyze and extend LFEA to LL-LFEA, and propose a countering method that can use null space invariance for watermark verification. We conduct an overall design from three aspects: watermark generation, watermark embedding, and watermark verification. Two indicators, WER and NSMD, are used to jointly verify the existence of watermark and identity of the model owner. Experiments demonstrate the effectiveness, scalability, reliability, and fidelity of NSmark, and it performs satisfactorily under various attacks. With the cooperation of two verification methods, a robust and secure watermarking scheme works.

## Limitations

Our approach has limitations in two main aspects. First, our method only focuses on LMs as we use the vocab to build connection between $sig$ and trigger. Solutions applicable to other models (such as visual or multi-modal models) need further exploration. Second, further research is needed on this basis to further protect the downstream network.

## Ethical Considerations

This work aims to improve the safety of LMs by adding more robust watermark. While we describe attack methods for completeness, our focus is defensive. We encourage responsible use of our detection tools.

## 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.

Sheldon Axler. 2015. *Linear algebra done right*. Springer.

Tony Cai, Jianqing Fan, and Tiefeng Jiang. 2013. Distributions of angles in random packing on spheres. *The Journal of Machine Learning Research*, 14(1):1837–1864.

Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. 2024. Stealing part of a production language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 5680–5705.

Huajie Chen, Chi Liu, Tianqing Zhu, and Wanlei Zhou. 2024. When deep learning meets watermarking: A survey of application, attacks and defenses. *Computer Standards & Interfaces*, page 103830.

Lu Chen, Shaofeng Li, Benhao Huang, Fan Yang, Zheng Li, Jie Li, and Yuan Luo. 2025. Contrasting adversarial perturbations: The space of harmless perturbations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 2114–2122.

Tianshuo Cong, Xinlei He, and Yang Zhang. 2022. Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 579–593.

Le Feng and Xinpeng Zhang. 2020. Watermarking neural network with compensation mechanism. In *Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30, 2020, Proceedings, Part II 13*, pages 363–375. Springer.

Weitao Feng, Wenbo Zhou, Jiyan He, Jie Zhang, Tianyi Wei, Guanlin Li, Tianwei Zhang, Weiming Zhang, and Nenghai Yu. 2024. AquaLoRA: Toward white-box protection for customized stable diffusion models via watermark LoRA. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 13423–13444.

Wendell Fleming. 2012. *Functions of several variables*. Springer Science & Business Media.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4115–4129.

Geoffrey Hinton. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. 2022. Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2043–2059. IEEE.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2.

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.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2024. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

Bowen Li, Lixin Fan, Hanlin Gu, Jie Li, and Qiang Yang. 2022. Fedipr: Ownership verification for federated deep neural network models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4521–4536.

Fang-Qi Li, Shi-Lin Wang, and Alan Wee-Chung Liew. 2023a. Linear functionality equivalence attack against deep neural network watermarks and a defense method by neuron mapping. *IEEE Transactions on Information Forensics and Security*, 18:1963–1977.

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

Yue Li, Hongxia Wang, and Mauro Barni. 2021. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294. Springer.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. 2022. Sok: How robust is image classification deep neural network watermarking? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 787–804. IEEE.

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

Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Benjamin Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. 2023. Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Wenjie Qu, Wengrui Zheng, Tianyang Tao, Dong Yin, Yanze Jiang, Zhihua Tian, Wei Zou, Jinyuan Jia, and Jiaheng Zhang. 2025. Provably robust multi-bit watermarking for {AI-generated} text. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 201–220.

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

Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D Spyropoulos, and Panagiotis Stamatopoulos. 2003. A memory-based approach to anti-spam filtering for mailing lists. *Information retrieval*, 6:49–73.

Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. 2024. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. *arXiv preprint arXiv:2405.04825*.

Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. 2021. Backdoor pre-trained models can transfer to all. *arXiv preprint arXiv:2111.00197*.

Anudeex Shetty, Yue Teng, Ke He, and Qiongkai Xu. 2024a. Warden: Multi-directional backdoor watermarks for embedding-as-a-service copyright protection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13430–13444.

Anudeex Shetty, Qiongkai Xu, and Jey Han Lau. 2024b. Wet: Overcoming paraphrasing vulnerabilities in embeddings-as-a-service with linear transformation watermarks. *arXiv preprint arXiv:2409.04459*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tianhao Wang and Florian Kerschbaum. 2019. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. IEEE.

Tianhao Wang and Florian Kerschbaum. 2021. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the Web Conference 2021*, pages 993–1004.

Yutong Wu, Han Qiu, Tianwei Zhang, Jiwei Li, and Meikang Qiu. 2022. Watermarking pre-trained encoders in contrastive learning. In *2022 4th International Conference on Data Intelligence and Security (ICDIS)*, pages 228–233. IEEE.

Hengyuan Xu, Liyao Xiang, Xingjun Ma, Borui Yang, and Baochun Li. 2024. Hufu: A modality-agnositc watermarking system for pre-trained transformers via permutation equivariance. *arXiv preprint arXiv:2403.05842*.

Zhilin Yang. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. 2023. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Machine Intelligence Research*, 20(2):180–193.

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

Renjie Zhu, Xinpeng Zhang, Mengte Shi, and Zhenjun Tang. 2020. Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing*, 2020:1–12.

# Appendix

## Table of Contents


**A The generation of Q** **12**

**B Proofs of Null Space Verification Theory** **12**

**C Estimation of NSMD** **12**

**D Watermark Generation** **13**

D.1 Trigger Generation Algorithm . 13

D.2 Verification Set Construction . 13

D.3 Process of Spread Spectrum Modulation and Despread Spectrum . . . . . 14

D.4 Spread Spectrum Modulation. . 14

D.5 Despread Spectrum. . . . . 14

**E Detailed Experimental Setup** **15**

**F LL-LFEA attack results** **15**

**G Reliability** **15**

**H Defense against LL-LFEA+finetuning** **17**

**I Robustness against fine-tuning attack** **17**

**J Robustness against multi-time LL-LFEA attack** **17**

**K Robustness against pruning** **17**

**L Robustness against Fine-pruning** **18**

**M Selection of Verification Set** **18**

**N Scalability of NSmark** **18**

**O Computational Cost Analysis** **18**

**P Feature Visualization** **18**

**Q The Use of Large Language Models (LLMs)** **19**


## A The generation of Q

In principle, $Q$ needs to be a reversible matrix. To make the value of $Q$ more stable for tensor calculations, we choose to constrain each element to be uniformly distributed between $[0, 1]$. The method to generate such a $Q$ is very simple, just sample uniformly between $[0, 1]$, and the resulting matrix is a reversible matrix (probability very close to 100%). The probability that the random matrix $Q$ is singular is the same as a point in $R^{n^2}$ lands in the zero set of a polynomial, which has Lebesgue measure 0 (Fleming, 2012).

| Method | Identity Binding | LL-LFEA Robustness |
|---|---|---|
| EmbMarker | ✗ | ✗ |
| WARDEN | ✗ | ✗ |
| WET | ✗ | ✗ |
| Hufu | ✗ | ✗ |
| **NSmark** | ✓ | ✓ |

Table 6: Comparison with existing watermark schemes.

## B Proofs of Null Space Verification Theory

*Proof.* The null space $N(A)$ of the matrix $A_{(a\times b)}$ is the set of all $b$-dimensional vectors $x$ that satisfy $Ax = \vec{0}$ (Axler, 2015). That is, $N(A) = \{x \in \mathbb{R}^b, Ax = \vec{0}\}$. Using $f_{wm}$ to denote the LM embedded with the watermark, assuming $A_{1(n\times m)} = \{f_{wm}(x), x \in D_T\}$ is the matrix concatenated from the output vectors, where $D_T$ is the verification dataset with watermark trigger, $m$ is the size of $D_T$ and $n$ is the dimension of the output vector of the last layer of the LM. Let the null space matrix of $A_1$ be $N_1$, then $A_1 \times N_1 = \mathbf{0}$.

After performing LL-LFEA, assuming that the new output matrix of $f_{wm}(D_T)$ is $A_{2(n\times m)}$, according to Section 3, we have $A_2 = Q \times A_1$. Then $A_2 \times N_1 = (Q \times A_1) \times N_1 = Q \times (A_1 \times N_1) = \mathbf{0}$, which means that $N_1$ belongs to the null space matrix of $A_2$. As $Q$ is a reversible matrix, then $rank(A_1) = rank(A_2)$, and the null spaces of $A_1$ and $A_2$ have the same dimension. It can be concluded that $N_1$ is also the null space matrix of $A_2$. □

## C Estimation of NSMD

We define NSMD by introducing the distribution of elements in a matrix, which is obtained by matrix multiplication of the output matrix $A$ of any LM without watermark and the null space matrix $N$ of $f_{wm}$. In $H_{(n\times p)} = A_{(n\times m)} \times N_{(m\times p)}$, $H_{j,j} = \alpha_i \cdot \beta_j$ is the dot product of the i-th row vector of $A$ and the j-th column vector of $N$. The approximate distribution of the angle between $n$ random uniformly distributed unit vectors in space

$\mathbb{R}^m$ is studied in (Cai et al., 2013). In space $\mathbb{R}^m$, given two random vectors uniformly distributed on the unit sphere, the angle $\theta$ between the two random vectors converges to a distribution whose probability density function is:

$$f(\theta) = \frac{1}{\sqrt{\pi}} \cdot \frac{\Gamma(\frac{m}{2})}{\Gamma(\frac{m-1}{2})} \cdot (\sin\theta)^{m-2}, \theta \in [0, \pi]. \quad (5)$$

When $m = 2$, $f(\theta)$ is uniformly distributed on $[0, \pi]$; when $m > 2$, $f(\theta)$ has a single peak at $\theta = \frac{\pi}{2}$. When $m > 5$, the distribution of $f(\theta)$ is very close to the normal distribution. Most of the $C_m^2$ angles formed by $m$ unit vectors randomly uniformly distributed are concentrated around $\frac{\pi}{2}$, and this clustering will enhance with the increase of the dimension $m$, because if $\theta \neq \frac{\pi}{2}$, then $(\sin\theta)^{m-2}$ will converge to 0 faster. This shows that in high-dimensional space, two randomly selected vectors are almost orthogonal.

We further derive the distribution of the dot product of two random vectors uniformly distributed and independently selected on the unit ball in space $\mathbb{R}^m$. Let $\alpha$ and $\beta$ be unit vectors and let $\theta$ be the angle between them, then $\alpha \cdot \beta = \cos(\theta)$. It is known that $\theta$ obeys the probability distribution $f(\theta)$, then the probability density function of $y = \alpha \cdot \beta = \cos(\theta), y \in [-1, 1]$ is:

$$\begin{aligned} g(y) =& g(\cos(\theta)) = f(\arccos(\cos(\theta))) \cdot \\ & |d(\arccos(\cos(\theta)))/d(\cos(\theta))|, \end{aligned} \quad (6)$$

where $d(\arccos(\cos(\theta)))/d(\cos(\theta)) = -1/\sqrt{(1-\cos^2(\theta))}$ is the derivative of the inverse cosine function. It can be inferred that:

$$g(y) = g(\cos(\theta)) = f(\theta)/\sqrt{(1-\cos^2(\theta))}. \quad (7)$$

Further, we analyze the mathematical expectation and variance of $Y = \cos(\Theta)$. The mean is:

$$\begin{aligned} EY &= \int_{-1}^{1} y \cdot g(y) dy \\ &= \int_{-1}^{1} y \cdot f(\arccos y)/\sqrt{(1-y^2)} dy. \end{aligned} \quad (8)$$

Note $k_m = \frac{1}{\sqrt{\pi}} \cdot \frac{\Gamma(\frac{m}{2})}{\Gamma(\frac{m-1}{2})}$, then:

$$EY = k_m \cdot \int_0^{\pi} \cos\theta \cdot (\sin\theta)^{m-2} d\theta = 0. \quad (9)$$

Its variance is:

$$\begin{aligned} DY = EY^2 - (EY)^2 = EY^2 &= \int_{-1}^{1} y^2 \cdot g(y) dy \\ &= k_m \cdot \int_0^{\pi} (\cos\theta)^2 \cdot (\sin\theta)^{m-2} d\theta \\ &= k_m \cdot \left( \int_0^{\pi} (\sin\theta)^{m-2} d\theta - \int_0^{\pi} (\sin\theta)^m d\theta \right) \\ &= \frac{2}{\sqrt{\pi}} \cdot \frac{\Gamma\left(\frac{m}{2}\right)}{\Gamma\left(\frac{m-1}{2}\right)} \cdot (I_{m-2} - I_m), \end{aligned} \quad (10)$$

where:

$$I_m = \int_0^{\pi/2} (\sin\theta)^m d\theta = \begin{cases} \frac{(m-1)!!}{m!!} \cdot \frac{\pi}{2} & m \text{ is even} \\ \frac{(m-1)!!}{m!!} & m \text{ is odd} \end{cases}. \quad (11)$$

As $m$ increases, $DY$ gradually approaches 0. Table 7 shows the specific values of the variance when $m$ takes specific values. Under the common output dimension of LM, that is, when $m = 1000$ or so, $DY$ is still a distance to 0.

Because the multiplication of the output matrix $A_1$ of the model embedded with watermark and its null space $N_1$ is exactly 0, while the variance of the elements obtained by the multiplication of the output matrix of other irrelevant models and $N_1$ is different from 0, we use and amplify this gap to define a new verification indicator - Null Space Match Degree (NSMD, defined in Equation 1) for watermark verification.

As $\sqrt{|h_{i,j}|} \in [0, 1]$ and $DY = 0$, we have

$$\begin{aligned} \text{NSMD}(A, N) &> \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{p} H_{i,j}^2 \\ &= p \cdot EY^2 \\ &= p \cdot (DY + (EY)^2) \\ &= p \cdot DY. \end{aligned} \quad (12)$$

Furthermore, $\text{NSMD}(A, N) > p \cdot DY$. For example, if $n = 768$ and $p = 1500$, we have NSMD $> 27.48$. If $N$ is the null space matrix of $A$, NSMD is a minimum value close to 0.

# D Watermark Generation

## D.1 Trigger Generation Algorithm

The trigger generation algorithm is shown in Algorithm 1.

## D.2 Verification Set Construction

Thus a candidate pool $D_{NS}$ (such as the original validation set) to generate null space verification

| $m$ | 10 | 20 | 300 | 768 | 1024 | 100000 |
|---|---|---|---|---|---|---|
| DY | 0.15667 | 0.11217 | 0.029302 | 0.018323 | 0.015870 | $7.1830 \times 10^{-6}$ |

Table 7: The value of DY in different dimensions $m$.

**Algorithm 1** Trigger Generation Algorithm

**Input**: owner's private key $K_{pri}$, identity information message $m$
**Output**: digital signature $sig$, trigger word $t$, verification set $D_V$

$sig \leftarrow$ **Sign**$(m, K_{pri})$.
$t \leftarrow$ **Encode**$(sig, n = 1)$
$sig_{sm} \leftarrow$ **SM**$(sig)$
$D_V \leftarrow$ **Select**$(sig)$
**return** $sig, sig_{sm}, t, D_V$

dataset should be published, and then samples are selected from the $D_{NS}$ as the verification data set $D_V$ based on the digital signature. We define the verification dataset selection algorithm as **Select**$(sig) \rightarrow D_V$, which must be a deterministic algorithm, that is, for the same input, there must be the same output. In addition, we hope that the algorithm will have different outputs for different inputs. Therefore, we choose a hash function and use a one-way hash chain to generate $D_V$. We hope that the index repetition rate obtained by different hash-value mappings is low, so we hope that the data set $D_{NS}$ is as large as possible. The specific process of the **Select**$(\cdot)$ algorithm is shown in Algorithm 2.

**Algorithm 2** Select Algorithm

**Input**: digital signature $sig$, $|D_V| = q$, candidate data pool $D_{NS}$
**Output**: verification set $D_V$

initialize $D_V \leftarrow [\,]$.
$h_0 \leftarrow$ **Hash**$(sig)$
**for** $i = 1$ to $q$ **do**
  $h_i \leftarrow$ **Hash**$(h_{i-1})$
  $idx_i \leftarrow h_i \% \, len(D_{NS})$
  $D_V$.append$(D_{NS}[idx_i])$
**end for**
**return** $D_V$

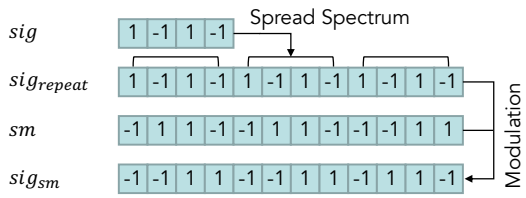


Figure 6: Example diagram of spread spectrum modulation. Repeat $sig$ to obtain $sig_{repeat}$, then use $sm$ to modulate $sig_{repeat}$ to obtain the spread spectrum modulated digital signature $sig_{sm}$.

### D.3 Process of Spread Spectrum Modulation and Despread Spectrum

### D.4 Spread Spectrum Modulation.

Assume that the digital signature $sig$ is $n$ bits, $sig = \{a_i | a_i \in \{-1, 1\}, i \in [0, n-1]\}$, and set the spread factor to $k$. Expand $sig$ horizontally by $k$ times to obtain $sig_{repeat} = \{ra_j | ra_j = a_i, i = j \bmod n, ra_j \in \{-1, 1\}, j \in [0, k \times n - 1]\}$. Input $sig$ as a seed in the pseudo-random generator to obtain the key $sm = \{b_j | b_j \in \{-1, 1\}, j \in [0, k \times n - 1]\}$ for spread spectrum modulation. Use $sm$ to modulate $sig_{repeat}$ to obtain the spread spectrum modulated digital signature $sig_{sm} = \{sa_j | sa_j = ra_j \times b_j, j \in [0, k \times n - 1]\}$. Figure 6 shows an example of $3\times$ spreading.

### D.5 Despread Spectrum.

Despread spectrum is the inverse process of the spread spectrum. Let the output of the mapping vector by $E$ be $O = \{o_j, j \in [0, k \times n - 1]\}$, modulate it with $sm = \{b_j | b_j \in \{-1, 1\}, j \in [0, k \times n - 1]\}$ to get $O_{repeat} = \{ro_j | ro_j = o_j / b_j, j \in [0, k \times n - 1]\}$, then quantify $O_{repeat}$ to get $O_{quan} = \{qo_j | qo_j \in \{-1, 0, 1\}, j \in [0, k \times n - 1]\}$. Finally, the signature is extracted by counting the number of $n$ positions that appear most often in $k$ copies. $sig' = \{a'_i | a'_i \in \{-1, 0, 1\}, i \in [0, n-1]\}$. The quantification method is shown as follows:

$$qo_j = \begin{cases} 1 & , 0.5 < ro_j < 1.5 \\ -1 & , -1.5 < ro_j < -0.5. \\ 0 & , \text{otherwise} \end{cases} \quad (13)$$

At last the signature is extracted as $sig' = \{a'_i | a'_i \in \{-1, 0, 1\}, i \in [0, n-1]\}$.

## E Detailed Experimental Setup

**Datasets.** We use WikiText-2 (Merity et al., 2017) as the watermark embedding dataset. To evaluate whether watermarking affects downstream utility, we consider five classification datasets covering different application scenarios: SST-2 and SST-5 (Socher et al., 2013) for sentiment analysis, Lingspam (Sakkis et al., 2003) for spam detection, OffensEval (Zampieri et al., 2019) for offensive language identification, and AG News (Zhang et al., 2015) for news classification.

**Models.** The main experiments use the base versions of BERT (Kenton and Toutanova, 2019), RoBERTa (Liu, 2019), DeBERTa (He et al., 2020), and XLNet (Yang, 2019). To further evaluate scalability, we also conduct supplementary experiments on BERT-large, RoBERTa-large, GPT-2 (Radford et al., 2019), and Llama-2-7B (Touvron et al., 2023). All pretrained weights are obtained from HuggingFace.

**Extractor architecture.** The extractor $E$ is implemented as a three-layer linear network. The two hidden layers contain 2048 and 1024 neurons, respectively. The input dimension of $E$ matches the output dimension of the corresponding LM, and the output dimension matches the length of the spread-spectrum signature $sig^{sm}$.

**Watermark configuration.** We use an identity-related string as the ownership message $msg$, such as "BERT is proposed by Google in 2018". The digital signature length is set to 256 bits. We use a spread factor of $k = 3$, producing a 768-bit spread-spectrum signature. SST-2 is used as the candidate pool $D_{NS}$ for constructing the verification set, and the verification set size is set to $q = 1500$. For the trigger set, the trigger is inserted into random positions five times.

**Training details.** For watermark embedding, we set $\lambda_1 = 0.5$ in $L_{\text{Extractor}}$ and $\lambda_2 = 0.2$ in $L_{\text{wm}}$. The batch size is 4, and the learning rates of both the watermarked model $f_{\text{wm}}$ and the extractor $E$ are set to $10^{-4}$. The two components are trained alternately for 10 epochs. For downstream fine-tuning, we use a learning rate of $2 \times 10^{-5}$, a batch size of 8, and train for 3 epochs.

**Evaluation metrics.** We use accuracy (ACC) to evaluate downstream model utility. For watermark verification, we report Watermark Extraction Rate (WER), which measures whether the embedded signature can be correctly extracted, and Null Space Matching Degree (NSMD), which measures whether the suspicious model output matches the registered null-space key.

## F LL-LFEA attack results

Table 8 shows the results of the LL-LFEA attack on EmbMarker (Peng et al., 2023) and WARDEN (Shetty et al., 2024a). As defined in (Peng et al., 2023), the difference in cosine similarity ($\Delta_{cos}$), the difference in squared L2 distance ($\Delta_{l2}$) and the p-value of the KS test are used to measure the effectiveness of the watermark. RedAlarm (Zhang et al., 2023) is another baseline watermark that demonstrates the effectiveness of EmbMarker and WARDEN. After the LL-LFEA attack, all the metrics of EmbMarker and WARDEN change a lot, and $\Delta_{cos}$ and $\Delta_{l2}$ are very close to those of the clean model, which fully demonstrates the effectiveness of LL-LFEA on existing watermarking schemes. Table 9 and 10 further elucidate the robustness and vulnerabilities inherent in WET (Shetty et al., 2024b) and Hufu (Xu et al., 2024) against LL-LFEA.

At the same time, LL-LFEA has little degradation on original model performance. This fully demonstrates the effectiveness of LL-LFEA on existing watermarking schemes.

Table 11 and Table 12 further prove the fidelity of LL-LFEA. Regardless of the model w/ or w/o watermark, the LL-LFEA attack has almost no impact on the subsequent accuracy.

## G Reliability

The watermark key is a triple $key = (sig, E, N)$. Next, we analyze whether the watermark can be successfully verified if an attacker provides an incorrect $key$.

• **Wrong signature** $sig$**.** The trigger $t$ and the output of $f_{wm}$ are related to $sig$. Next, we will study whether the watermark can be verified normally when some of these factors are wrong. Figure 7 shows all possible scenarios.

For $f_{wm}$, (i) when the trigger is wrong ($t_w$) and the signature is correct ($sig_c$), WER = 0, NSMD $> T_N$. This means that $f_{wm}$ has learned the relationship between $sig$ and $t$. Whether $t$ is correct

| Dataset | Method | ACC | p-value ↓ | $\Delta_{cos}$ ↑ | $\Delta_{l2}$ ↓ |
|---|---|---|---|---|---|
| SST2 | None | 93.76 | >0.34 | -0.07 | 0.14 |
| | RedAlarm | 93.76 | >0.09 | 1.35 | -2.70 |
| | EmbMarker | 93.55 | $<10^{-5}$ | 4.07 | -8.13 |
| | WARDEN | 92.89 | $<10^{-11}$ | 8.06 | -16.12 |
| | **EmbMarker+LL-LFEA** | **92.43** | **0.01** | **0.14** | **-0.28** |
| | **WARDEN+LL-LFEA** | **93.12** | $>10^{-5}$ | **0.34** | **-0.69** |
| MIND | None | 77.30 | >0.08 | -0.76 | 1.52 |
| | RedAlarm | 77.18 | >0.38 | -2.08 | 4.17 |
| | EmbMarker | 77.29 | $<10^{-5}$ | 4.64 | -9.28 |
| | WARDEN | 77.22 | $<10^{-10}$ | 7.49 | -14.99 |
| | **EmbMarker+LL-LFEA** | **75.08** | **0.01** | **-0.70** | **1.39** |
| | **WARDEN+LL-LFEA** | **77.17** | $>10^{-7}$ | **0.51** | **-1.03** |
| AGNews | None | 93.74 | >0.03 | 0.72 | -1.46 |
| | RedAlarm | 93.74 | >0.09 | -2.04 | 4.07 |
| | EmbMarker | 93.66 | $<10^{-9}$ | 12.85 | -25.70 |
| | WARDEN | 93.25 | $<10^{-10}$ | 9.82 | -19.63 |
| | **EmbMarker+LL-LFEA** | **91.86** | **0.005** | **0.48** | **-0.96** |
| | **WARDEN+LL-LFEA** | **93.47** | $>10^{-10}$ | **0.87** | **-1.74** |
| Enron | None | 94.74 | >0.03 | -0.21 | 0.42 |
| | RedAlarm | 94.87 | >0.47 | -0.50 | 1.00 |
| | EmbMarker | 94.78 | $<10^{-6}$ | 6.17 | -12.34 |
| | WARDEN | 94.65 | $< 10^{-7}$ | 5.1 | -10.2 |
| | **EmbMarker+LL-LFEA** | **92.40** | **0.01** | **0.19** | **-0.39** |
| | **WARDEN+LL-LFEA** | **94.85** | $>10^{-6}$ | **0.47** | **-0.94** |

Table 8: Results of *LL-LFEA* attack on *EmbMarker* and *WARDEN*. For watermark, ↑ means higher metrics are better. ↓ means lower metrics are better. In contrast, after *LL-LFEA* attack, the higher *p-value*, $\Delta_{l2}(\%)$ and lower $\Delta_{cos}(\%)$ can illustrate the effectiveness of *LL-LFEA*.

| Dataset | Method | AUC | p-value ↓ | $\Delta_{cos}$ ↑ | $\Delta_{l2}$ ↓ |
|---|---|---|---|---|---|
| SST2 | WET | 100.00 | $1.45 \times 10^{-11}$ | 79.22 | -158.45 |
| | **WET+LL-LFEA** | **94.25** | $1.33 \times 10^{-6}$ | **1.46** | **-4.90** |
| MIND | WET | 100.00 | $1.45 \times 10^{-11}$ | 76.74 | -153.49 |
| | **WET+LL-LFEA** | **20.25** | **0.01** | **-1.52** | **3.04** |
| AGNews | WET | 100.00 | $1.45 \times 10^{-11}$ | 81.69 | -163.36 |
| | **WET+LL-LFEA** | **58.25** | **0.57** | **0.35** | **-0.69** |
| Enron | WET | 100.00 | $1.45 \times 10^{-11}$ | 81.88 | -163.76 |
| | **WET+LL-LFEA** | **49.50** | **0.83** | **0.04** | **0.07** |

Table 9: Results of *LL-LFEA* attack on *WET*. For watermark, ↑ means higher metrics are better. ↓ means lower metrics are better. The distinct changes in *p-value*, $\Delta_{cos}(\%)$ and $\Delta_{l2}(\%)$ after attack demonstrate the effectiveness of *LL-LFEA*.

| Dataset | | Method | ACC | WR |
|---|---|---|---|---|
| Watermark | Downstream | | | |
| SST-2 | IMDB | Hufu | 93.80 | 100.00 |
| | | **Hufu+LL-LFEA** | **93.47** | **0.00** |
| SST-2 | QNLI | Hufu | 86.62 | 100.00 |
| | | **Hufu+LL-LFEA** | **87.90** | **0.00** |
| AG News | SST-2 | Hufu | 91.17 | 100.00 |
| | | **Hufu+LL-LFEA** | **91.16** | **0.00** |
| IMDB | QNLI | Hufu | 89.42 | 100.00 |
| | | **Hufu+LL-LFEA** | **87.86** | **0.00** |

Table 10: Results of *LL-LFEA* attack on *Hufu*. Different watermark datasets and downstream datasets are evaluated in Hufu.

| Model | LL-LFEA | SST-2 | SST-5 | Offenseval | Lingspam | AGNews |
|---|---|---|---|---|---|---|
| BERT | ✗ | 91.63 | 53.03 | 84.07 | 99.66 | 94.38 |
| | ✓ | 92.20 | 51.72 | 82.44 | 99.66 | 94.24 |
| RoBERTa | ✗ | 94.04 | 56.15 | 84.88 | 100.00 | 94.72 |
| | ✓ | 94.04 | 54.71 | 82.09 | 99.14 | 94.61 |
| DeBERTa | ✗ | 93.58 | 57.65 | 85.12 | 99.31 | 94.84 |
| | ✓ | 93.46 | 53.89 | 83.91 | 99.66 | 93.37 |
| XLNet | ✗ | 93.58 | 53.62 | 84.65 | 99.31 | 94.07 |
| | ✓ | 92.09 | 52.85 | 83.14 | 99.14 | 94.28 |

Table 11: For models without any watermark (fine-tune pretrained model on downstream task), the ACCs before and after LL-LFEA have no significant change.

| Model | LL-LFEA | SST-2 | SST-5 | Offenseval | Lingspam | AGNews |
|---|---|---|---|---|---|---|
| BERT | ✗ | 91.40 | 52.62 | 85.12 | 99.14 | 93.75 |
| | ✓ | 92.55 | 52.21 | 84.19 | 98.97 | 94.01 |
| RoBERTa | ✗ | 92.55 | 54.71 | 84.30 | 99.66 | 94.43 |
| | ✓ | 92.55 | 53.27 | 84.88 | 99.31 | 94.34 |
| DeBERTa | ✗ | 93.00 | 55.48 | 83.02 | 99.31 | 94.61 |
| | ✓ | 93.00 | 51.95 | 82.91 | 99.31 | 93.97 |
| XLNet | ✗ | 88.65 | 42.67 | 81.98 | 99.14 | 93.29 |
| | ✓ | 88.53 | 42.58 | 79.95 | 99.48 | 93.24 |

Table 12: For models with NSmark(embed NSmarkusing WikiText-2, and finetune the whole model on downstream task), the ACCs before and after LL-LFEA have no significant change.

determines whether $f_{wm}$ can produce the expected output, which in turn affects both the calculation of WER and NSMD. (ii) When the trigger is correct ($t_c$) and the signature is wrong (considering the most dangerous scenario that $sig_w$ consists only of $\{-1, 1\}$), WER $\approx 0.5$. This is because $t_c$ leads to the right $sig'$, and its expectation of WER with a random string $\{-1, 1\}$ is 0.5. As the output matrix is correctly generated by $f_{wm}$ based on $t_c$, NSMD = 0 in this case. (iii) When both trigger and signature are wrong ($t_w$ and $sig_w$), WER = 0, NSMD$> T_N$, indicating that the watermark cannot be correctly verified without providing the correct key. (iv) For a model without embedded watermarks $f_{clean}$, watermarks cannot be extracted even if the correct key is provided.

• **Wrong extractor $E$.** Since $E$ is not involved in NSMD calculation, we only analyze the impact of $E$ on WER. As shown in Table 13, when $E$ is wrong, the WER is close to 0, indicating that wrong $E$ is unable to extract the watermark.

• **Wrong null space $N$.** Since $N$ is not used in calculating the WER, we only analyze its impact on NSMD. As shown in Table 14, $N_r$ is a randomly

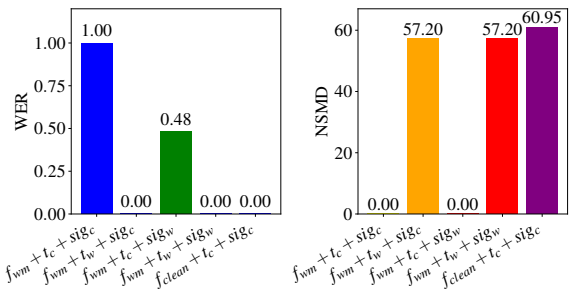


Figure 7: The impact of the correctness of trigger $t$ and signature $sig$ on WER and NSMD. $c$ and $w$ in the subscript stands for *correct* and *wrong*. Only $f_{wm}$ with correct trigger and $sig$ can pass through verification.

| Setting | BERT | RoBERTa | DeBERTa | XLNet |
|---|---|---|---|---|
| $f_{wm} + E_c$ | 1.00 | 1.00 | 1.00 | 1.00 |
| $f_{wm} + E_w$ | 0.00 | 0.00 | 0.00 | 0.13 |

Table 13: WER results of different extractor $E$. $E_c$, and $E_w$ means correct $E$ and wrong $E$, respectively. Only the correct $E$ can extract the signature.

generated matrix with the same dimension as $N$ and each element is distributed between $[0, 1]$. It can be seen that NSMD is very large at this time. However, if the attacker knows the watermark algorithm and wants to reduce NSMD, a $N_s$ with extremely small elements might be generated. In this case, NSMD might meet the verification requirements. This indicates that NSMD cannot be used independently to verify the watermark.

## H Defense against LL-LFEA+finetuning

After applying LL-LFEA, the attacker may add a network to $f_{LL-LFEA}$ and fine-tune it for downstream tasks. We hope the model after the LL-LFEA+fine-tuning attack can still maintain the watermark. Table 15 shows results on different LMs and different downstream tasks are not exactly the same. WER of different models has decreased significantly to varying degrees. Most NSMDs are still below the threshold, but RoBERTa and DeBERTa exhibit greater variance on SST-2, which is

| Setting | BERT | RoBERTa | DeBERTa | XLNet |
|---|---|---|---|---|
| $N_c$ | $2.94 \times 10^{-6}$ | $2.53 \times 10^{-6}$ | $2.91 \times 10^{-6}$ | $2.90 \times 10^{-6}$ |
| $N_r$ | 3167.81 | 3171.58 | 3182.79 | 3117.61 |
| $N_s$ | 1001.75 | 1002.94 | 1006.49 | 985.87 |

Table 14: NSMD of different null space matrix $N$. $N_c$, $N_r$ and $N_s$ means correct $N$, random $N$ and $N$ composed of small elements, respectively. Only when the correct $N$ is used can the NSMD be close to 0.

| Metric | Model | SST-2 | SST-5 | Offenseval | Lingspam | AGnews |
|---|---|---|---|---|---|---|
| WER | BERT | 0.29 | 0.29 | 0.28 | 0.31 | 0.37 |
| | RoBERTa | 0.47 | 0.07 | 0.07 | 0.35 | 0.35 |
| | DeBERTa | 0.30 | 0.28 | 0.29 | 0.29 | 0.42 |
| | XLNet | 0.00 | 0.00 | 0.00 | 0.01 | 0.01 |
| NSMD | BERT | 15.43 | 15.06 | 12.95 | 12.56 | 13.17 |
| | RoBERTa | 47.39 | 17.96 | 14.64 | 12.73 | 28.89 |
| | DeBERTa | 26.86 | 17.51 | 20.38 | 20.77 | 38.38 |
| | XLNet | 13.20 | 12.08 | 14.12 | 12.98 | 14.37 |
| ACC | BERT | 91.17 | 52.22 | 86.04 | 99.48 | 93.80 |
| | RoBERTa | 93.00 | 52.71 | 84.88 | 99.14 | 94.37 |
| | DeBERTa | 94.04 | 51.95 | 82.91 | 99.48 | 93.70 |
| | XLNet | 90.14 | 42.40 | 81.40 | 99.48 | 93.03 |

Table 15: Impact of LL-LFEA+ fine-tuning attack on watermark. (i) WER increases to varying degrees; (ii) NSMD are still below the threshold; (iii) ACC does not decrease significantly.

| # LL-LFEA Times | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| WER | 1.00 | 0.27 | 0.00 | 0.00 |
| NSMD | $2.94 \times 10^{-6}$ | 0.06 | 0.04 | 0.05 |

Table 16: The results of multi-time LL-LFEA attack on watermark performance.

generally consistent with that of fine-tuning without LL-LFEA attack (Table 2). Through ACC, we can find that LL-LFEA attack does not affect the performance of the model on the original task.

## I Robustness against fine-tuning attack

Figure 8 further illustrates the effect of different learning rates used in fine-tuning. The results indicate that learning rates have little impact on ACC and WER, maintaining watermark robustness. Similar to the results above, NSMD exhibits relatively more instability. However, the combined validation mechanism of the two metrics ensures that WER alone can still accurately validate the ownership of the model.

## J Robustness against multi-time LL-LFEA attack

Since LL-LFEA has little damage on the model performance, the attacker may try to further destroy the watermark through multiple LL-LFEA attacks. In principle, multiple LL-LFEA will only decrease WER but will not affect NSMD. Table 16 shows the results consistent with our analysis.

## K Robustness against pruning

Table 17 shows results that the watermarked model is pruned and finetuned on the SST-5 dataset. The

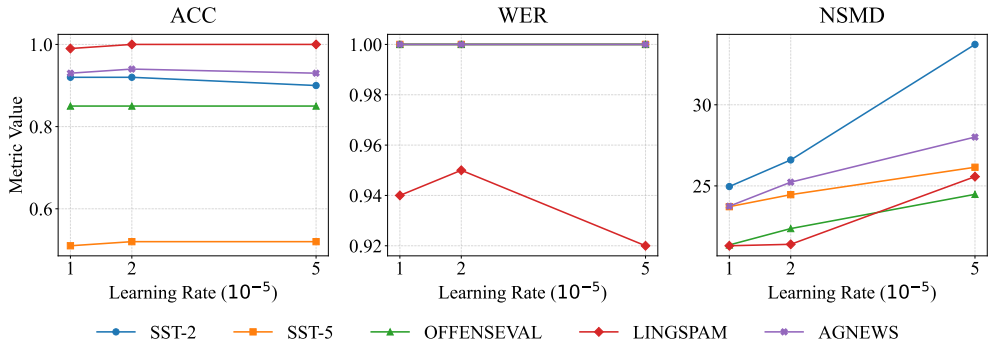


Figure 8: Impact of learning rate selection during fine-tuning attack.

accuracy of the model only changes slightly.

## L Robustness against Fine-pruning

Table 18 uses SST-5 as the fine-tuning dataset to show the watermark extraction effect after fine-pruning. It can be seen that the results are generally the same as Figure 5.

## M Selection of Verification Set

We explore the necessity of using trigger set in verification rather than clean set. As shown in Table 19, before downstream fine-tuning $(f_{wm})$, NSMD for trigger set ($\text{NSMD}_t$) and clean set ($\text{NSMD}_c$) are all close to 0.After downstream fine-tuning, $\text{NSMD}_c$ is significantly higher than $\text{NSMD}_t$.

Combining these results, we think fine-tuning has little effect on the output representation of trigger set. However, the output representation of the clean set will change significantly for better performance on different downstream tasks, which causes the null space matrix no longer match the original $N$. Thus $\text{NSMD}_c$ has significantly increase. Therefore, the trigger set is needed for verifying null space.

## N Scalability of NSmark

We further test the scalability of NSmarkon larger models, including BERT-large-uncased,[1] RoBERTa-large,[2] GPT-2,[3] and Llama-2-7B[4]. Experimental results in Table 20 demonstrate the scalability and effectiveness of NSmarkacross different size of models.

[1]https://huggingface.co/google-bert/bert-large-uncased
[2]https://huggingface.co/FacebookAI/roberta-large
[3]https://huggingface.co/openai-community/gpt2
[4]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

## O Computational Cost Analysis

The computational cost of NSmarkis acceptable compared to existing methodologies. Specifically, the cost encompasses both model-related and non-model-related components. Model-related computations, such as extractor training (a three-layer MLP) and reference model support (a replica of the original LM used solely for inference), are exclusively utilized in watermark embedding. This embedding is executed once per model and occurs on the model training side, requiring substantial computational power. Non-model-related computations involve extensive mathematical and cryptographic processes, including signature and hash algorithm calculations. These cryptographic computations incur negligible costs on current computing devices, rendering watermark theft computationally impractical. In contrast, other existing approaches, such as WET (Shetty et al., 2024b), necessitate the generation of extensive data via ChatGPT (costing over $100) or DIPPER (Krishna et al., 2024) (11B model), which demands significantly more time, computation, and financial resources than NSmark. We evaluated the cost of NSmarkusing a single Nvidia RTX 3090. The model employed was BERT, and other configurations were consistent with Section 5.1. Table 21 presents the time cost results (s: second, h: hour). `Model_Emb` refers to the total duration of watermark model training, while `Original_Model_Train` denotes the training time for a model without a watermark. These findings suggest that NSmarkis practical for real-world applications.

## P Feature Visualization

To demonstrate the effectiveness of our scheme, we use t-SNE to visualize the feature distribution of

| **Pruning Rate** | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| ACC(%) | 52.62 | 52.36 | 52.13 | 51.99 | 52.35 | 51.62 | 51.94 | 51.71 | 50.81 | 47.23 |

Table 17: Results of different pruning rates on ACC of watermarked model.

| **Pruning Rate** | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | $f_{clean}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ACC | 52.62 | 52.35 | 51.11 | 51.99 | 52.35 | 51.63 | 52.94 | 52.71 | 50.81 | 51.44 | 53.03 |
| WER | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.67 | 0.00 |
| NSMD | 25.29 | 20.13 | 18.45 | 20.76 | 21.18 | 21.19 | 25.37 | 28.78 | 42.07 | 50.81 | 70.06 |

Table 18: Impact of fine-pruning attack on watermark. Generally the results are similar to results of pruning attack in Figure 5.

| **Dataset** | **Metric** | **BERT** | **RoBERTa** | **DeBERTa** | **XLNet** |
|---|---|---|---|---|---|
| SST-2 | $\text{NSMD}_t$ | 29.77 | 50.17 | 31.89 | 24.12 |
| | $\text{NSMD}_c$ | 76.20 | 74.37 | 74.31 | 68.86 |
| SST-5 | $\text{NSMD}_t$ | 25.29 | 30.97 | 25.74 | 23.52 |
| | $\text{NSMD}_c$ | 73.39 | 71.75 | 71.83 | 55.43 |
| Offenseval | $\text{NSMD}_t$ | 22.52 | 25.15 | 23.52 | 25.29 |
| | $\text{NSMD}_c$ | 64.55 | 64.50 | 64.48 | 41.23 |
| Lingspam | $\text{NSMD}_t$ | 21.96 | 26.78 | 27.23 | 24.06 |
| | $\text{NSMD}_c$ | 66.60 | 65.90 | 50.73 | 24.88 |
| AGnews | $\text{NSMD}_t$ | 24.37 | 28.43 | 37.69 | 26.20 |
| | $\text{NSMD}_c$ | 74.75 | 74.95 | 73.67 | 38.02 |

Table 19: The necessity of using trigger set in verification. $\text{NSMD}_t$ is the result for trigger set and $\text{NSMD}_c$ is for clean set.

the watermarking model. As shown in Figure 9, the input with trigger and the input without trigger can be well separated in the output of LM and $E$, whether for $f_{wm}$ or fine-tuned $F_{wm}$.

## Q The Use of Large Language Models (LLMs)

We disclose that `Gemini-3-Pro` is used as a general-purpose writing assistant in the preparation of this paper. The LLMs' role is strictly limited to improving clarity, grammar, and style (i.e., to aid or polish writing). The human authors are fully responsible for all substantive content, claims, and conclusions presented in this paper, and have carefully reviewed and edited all text to ensure its scientific accuracy and integrity.

| Metric | $f_{wm}$ | | | | $f_{clean}$ | | | |
|---|---|---|---|---|---|---|---|---|
| | **BERT-large** | **RoBERTa-large** | **GPT-2** | **Llama-2-7B** | **BERT-large** | **RoBERTa-large** | **GPT-2** | **Llama-2-7B** |
| WER | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.03 | 0.00 | 0.00 |
| NSMD | $2.08 \times 10^{-8}$ | $1.99 \times 10^{-6}$ | $3.29 \times 10^{-6}$ | $2.97 \times 10^{-6}$ | 72.59 | 71.22 | 80.33 | 82.94 |

Table 20: Effectiveness of watermark on larger LMs. Both WER and NSMD on larger models are similar to the main results, demonstrating the scalability of NSmark.

| Process | Sign | Encode | Select | SM | DSM | Gen_Q | Cal_NS | WM_Emb | Original_Model_Train | WM_Verify |
|---|---|---|---|---|---|---|---|---|---|---|
| Time | $10^{-4}$s | $10^{-5}$s | 10−3s | 0.5s | 0.03s | 0.27s | 0.83s | 4h | 3h | 37s |

Table 21: The time cost of NSmark. Results are tested on a single Nvidia RTX 3090. 's' denotes seconds and 'h' denotes hours. WM_Emb and Original_Model_Train means the time of watermark model training, and training model without watermark, respectively.

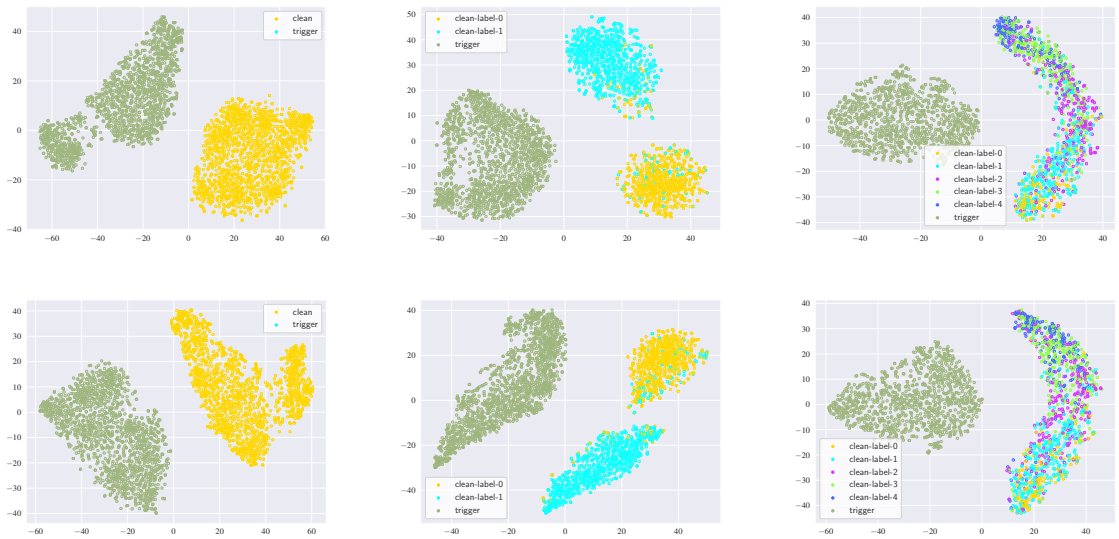


Figure 9: The t-SNE visualization of output feature vectors of watermarked models. (i) Left column: $f_{wm}$ on WikiText; (ii) Middle column: $F_{wm}$ on SST-2; (iii) Right column: $F_{wm}$ on SST-5.