NSMARK: Null Space Based Black-box Watermarking Defense Framework for Pre-trained Language Models

Anonymous ACL submission

Abstract

Pre-trained language models (PLMs) have emerged as critical intellectual property (IP) assets that necessitate protection. Although various watermarking strategies have been proposed, they remain vulnerable to Linear Functionality Equivalence Attack (LFEA), which can invalidate most existing white-box watermarks without prior knowledge of the watermarking scheme or training data. This paper further analyzes and extends the attack scenarios of LFEA to the commonly employed black-box settings for PLMs by considering Last-Layer outputs (dubbed LL-LFEA). We discover that the null space of the output matrix remains invariant against LL-LFEA attacks. Based on this finding, we propose NSMARK, a task-agnostic, black-box watermarking scheme capable of resisting LL-LFEA attacks. NSMARK consists of three phases: (i) watermark generation using the digital signature of the owner, enhanced by spread spectrum modulation for increased robustness; (ii) watermark embedding through an output mapping extractor that preserves PLM performance while maximizing watermark capacity; (iii) watermark verification, assessed by extraction rate and null space conformity. Extensive experiments on both pre-training and downstream tasks confirm the effectiveness, reliability, fidelity, and robustness of our approach. Code is available at https://anonymous.4open.science/r/NSmark-2FC1.

1 Introduction

During past decades, with the rapid development of artificial intelligence (AI), pre-trained language models (PLMs) have achieved superior performance and been applied in a wide range of fields. At the same time, training high-performance PLMs requires a large amount of data and computing resources, thus PLMs can be regarded as valuable intellectual property (IP). With the deployment and application of machine learning as a service (MLaaS) platforms, companies sell well-trained PLMs as commodities. Once sold models are illegally stolen, distributed or resold, the rights of the model owner will be seriously infringed. Therefore, it is necessary to protect IP of PLMs.

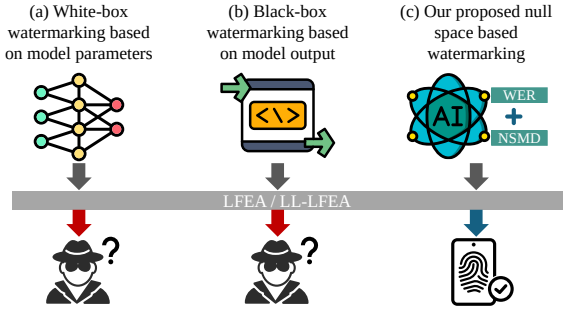

Figure 1: Illustration of different watermark schemes against LFEA/LL-LFEA.

Watermarking techniques have been widely applied for protecting IP of deep learning models (Chen et al., 2024; He et al., 2024). By incorporating identifiable information, these methods serve as a means to verify model ownership and provide proof of authenticity. Existing schemes can be divided into white-box and black-box categories according to whether the model parameters need to be accessed in verification. Among them, black-box schemes are more applicable when model parameters are inaccessible in most real-life scenarios, such as deployment as API.

However, protecting the IP of PLMs through watermarking presents significant challenges. Since PLMs can be deployed for various downstream tasks post-training (Zhang et al., 2023), it is crucial for watermark schemes to be task-independent. Besides, recent studies have revealed vulnerabilities and shortcomings in existing watermarking techniques (Li et al., 2023a). The proposed Linear Functionality Equivalence Attack (LFEA) is easy to conduct and can invalidate most existing white-box watermarks without knowledge of the

watermarking scheme or the training data by exploiting linear invariance. Considering Last-Layer outputs, we further analyze and expand the attack scenarios black-box settings using model outputs (dubbed LL-LFEA).

To address these problems, we first explore the characteristics of the model output. We find that the null space of the matrix composed of the output vectors of model is invariant under LL-LFEA. Based on this, we propose a new **null space verification method that can resist 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 protected PLM. Finally, we propose NSMARK, a null space based task-agnostic black-box watermarking scheme for PLMs. NSMARK uses identity information to generate all watermark-related elements, and uses WER and NSMD to verify the watermark. Spread spectrum modulation technology and an extra extractor are also introduced to enhance the watermark performance.

Our contributions are summarized as follows:

(i) We analyze the threat of LFEA on output-based watermark, and propose LL-LFEA, which can destroy the watermark embedded in the output vector without affecting the performance of downstream tasks.

(ii) We find that the null space of the matrix composed of the output vectors of model is invariant under LL-LFEA and thus propose a new null space verification method NSMARK that can resist LL-LFEA. Notably, NSMARK is task-agnostic that uses both new null space verification and signature verification to resist LL-LFEA.

(iii) We conduct comprehensive experiments by applying NSMARK to various models of both pre-training and downstream tasks. Experimental results demonstrate the effectiveness, fidelity, reliability and robustness of NSMARK.

2 Related Work

Watermarking for PLMs. With the rise of pre-training in NLP, recent work has explored watermarking specific to PLMs. BadPre (Jia et al., 2022) introduces a task-agnostic backdoor attack only for MLM-based PLMs. Hufu (Xu et al., 2024) introduces a modality-agnostic approach for pre-trained Transformer models using the permutation equivariance property. Explanation as a Watermark (Shao et al., 2024) addresses the limitations of backdoor-based techniques by embedding multi-bit watermarks into feature attributions using explainable AI. (Shen et al., 2021; Zhang et al., 2023) propose 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 didn't 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 vulnerable to attacks by LFEA or LL-LFEA in principle.

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 increased effectiveness. Knowledge Distillation (Hinton, 2015) techniques can also inadvertently remove watermarks while reducing model size. (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 the model's 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. Fraud attacks include overwriting (Wang and Kerschbaum, 2019) and ambiguity attacks (Zhu et al., 2020) also pose a great threat to watermarks.

3 Method

3.1 Threat Model

In white-box watermarking schemes, high-dimensional model parameters are often used as watermark information. For PLMs, since the output of the last layer is a high-dimensional vector, we can use a method similar to the white-box watermarking scheme to embed watermarks in the output. However, embedding identity information into the high-dimensional output vector will face the threat of LFEA-like attacks, which is proposed to destroy watermark information embedded in model parameters by linearly transforming parameters of

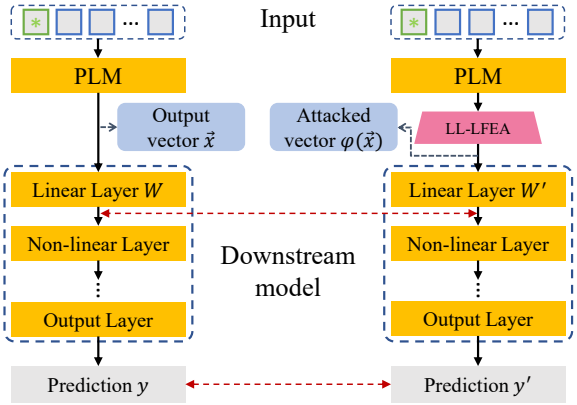

Figure 2: The schematic diagram of model inference flow before and after LL-LFEA attack.

intermediate layers. Next, we discuss the specific form of linear isomorphism attacks in this scenario.

Assuming that the attacker knows the watermark information is embedded in the output of the PLM, and hopes to destroy the watermark at a very low attack cost (without modifying the model structure or fine-tuning the model) without affecting the normal task performance of the model. As shown in Figure 2, we give an attack method that can meet this requirement and provide a proof below. Output vector $\vec{x}$ is obtained from PLM, which is also the input of downstream model. After a series of linear and nonlinear layers, the prediction result y is obtained. The attacker attempts to destroy the watermark by changing the output vector of PLM, but does not affect the final prediction result of the model, that is, the attacker hopes to change $\vec{x}$ to $\varphi(x)$ and input it into the downstream network, and corresponding prediction result $y' = y$. The sufficient condition for this result is that the modification of the PLM output vector can be offset after passing through the first linear layer of the downstream network. Let the parameter of the first linear layer of the downstream network be W, then the attacker hopes $W'\varphi(\vec{x}) = W\vec{x}$, then we have $\varphi(\vec{x}) = W'^{,\dagger}W\vec{x} = Q\vec{x}$, where $Q = W'^{,\dagger}W$ and $W'^{,\dagger}$ is the pseudo-inverse of W' (Li et al., 2023a). In order not to lose information in linear transformation (otherwise downstream tasks will be affected uncontrollably), Q must be a reversible matrix. This proves that attacker can perform a linear transformation on $\vec{x}$ to destroy the watermark embedded in the output vector without affecting the performance of downstream tasks. We call this attack the Last-Layer Linear Functionality Equivalence Attack (LL-LFEA).

3.2 Null Space Verification Theory

LL-LFEA performs linear transformation on output vector of PLM and can destroy the watermark embedded in it, which means previous watermark verification methods will be significantly affected. We find that the null space of the matrix composed of the output vector is invariant under the LL-LFEA attack, and propose to use the null space matching degree to verify whether the model is embedded with watermarks.

Theorem 1. *Before and after LL-LFEA, the null space of the output matrix of PLM remains unchanged for the same input set. (Proofs in Appendix A.1)*

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.

3.3 Null Space Match Degree (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 PLM without watermark and the null space matrix N of f_{wm}. In $H_{(n\times p)} = A_{(n\times m)} \times N_{(m\times p)}$, $H_{i,j} = \alpha_i \cdot \beta_j$ 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:

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

Furthermore, we give a detailed analysis of estimation of NSMD (in Appendix A.2). 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. This difference is amplified by the process of calculating the square root, 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.

3.4 Overall Framework of NSMARK

As shown in Figure 3, NSMARK scheme includes three modules: watermark generation, watermark embedding, and watermark verification:

3.4.1 Watermark Generation

The workflow of watermark generation is shown in Algorithm 1. We hope that the generated watermark contains the owner's identity information. First, the digital signature $sig = \textbf{Sign}(m)$ is generated from the identity information message m.

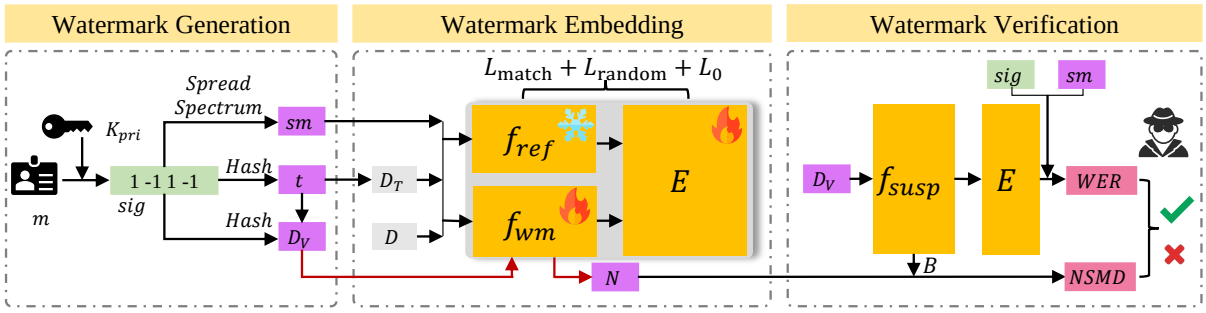

Figure 3: The overall workflow of NSMARK.

In order to ensure that the trigger t has a unique mapping relationship with the 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.

In order to defend against ambiguous attacks, the verification trigger set D_V used for null space verification also needs to be generated based on sig. A candidate pool D_{NS} for generating null space verification data sets needs to be published, and then a fixed number of samples are selected from the D_{NS} based on the digital signature as the verification data set D_V. We define the verification data set 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 has 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.

In order to improve the robustness of the watermark, we introduce spread spectrum modulation technology as (Feng and Zhang, 2020). Spread spectrum modulation technology uses redundant bits to represent the original information. Figure 4 shows an example of $3\times$ spreading. Please refer to the Appendix A.5.1 for the specific process **SM**$(sig) \rightarrow sig_{wm}$.

3.4.2 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. When taking $\{D, D_T\}$ as input, f_{wm} will output $\{V, V^T\}$ and f_{ref} will

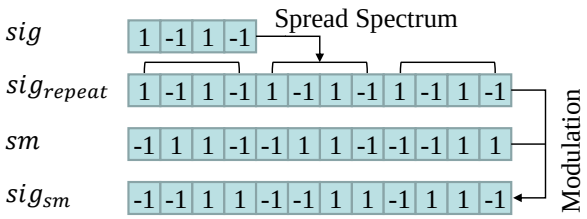

Figure 4: Example diagram of spread spectrum modulation.

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

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

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

We use cosine similarity as the sim function. The complete loss function of E is $L_{\text{Extractor}} = \lambda_1 L_{\text{match}} + (1 - \lambda_1) L_{\text{random}}$. When training E us-

ing the loss, 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 training loss function of PLM. During training, E and f_{wm} are trained alternately.

3.4.3 Watermark Verification

To better defend against possible attacks, NSMARK uses two metrics together to verify the ownership: Watermark Extracting Rate (WER) and NSMD. Model owner needs to submit $key = (sig, E, N)$ to Certification Authority (CA). CA generates t, D_V, sm using sig. Input D_V to 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 calculated after despread spectrum. Despread spectrum is the inverse process of spread spectrum (detailed process in Appendix A.5.2). At last the signature is extracted as $sig' = \{a'_i | a'_i \in \{-1, 0, 1\}, i \in [0, n-1]\}$. WER is defined as the proportion of sig and sig' having the same value:

$$\text{WER} = \frac{1}{n} \sum_{i=0}^{n-1} \left[a_i = a'_i \right] . \tag{4}$$

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

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

4 Experiments

4.1 Experimental Setup

Datasets. We use WikiText-2 (Merity et al., 2017) for pre-training and watermark embedding. To evaluate the performance on downstream tasks, we select many text classification datasets: 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. For PLMs, we use the base versions of BERT (Kenton and Toutanova, 2019), RoBERTa (Liu, 2019), DeBERTa (He et al., 2020) and XLNet (Yang, 2019). All pre-trained weights are from HuggingFace[1]. The extractor network is a three-layer linear network with hidden layers of 2048 and 1024 neurons. The input dimension matches the output dimension of the PLM. The output dimension matches the size of sig_{sm}.

Watermark settings and training details. We select a string containing owner information as the message m, e.g., "BERT is proposed by Google in 2018". The length of sig is 256 and is spread-spectrum with a spreading factor $k = 3$, resulting in a 768-bit sig_{sm}. We use the labeled downstream dataset SST-2 as the candidate pool D_{NS}. q, the length of D_V, is 1500. The trigger is inserted to random positions for 5 times in trigger set. When performing watermark embedding, $\lambda_1 = 0.5$ and $\lambda_2 = 0.2$ in $L_{Extractor}$ and L_{wm}. Batchsize is 4, and learning rates for both f_{wm} and E are 10^{-4}. f_{wm} and E are trained alternately for 10 epochs. When fine-tuning on downstream tasks, the learning rate is 2×10^{-5} and batchsize is 8 for 3 epochs.

Metrics. As mentioned before, We define two metrics to verify the model's identity: WER and NSMD. Besides, we adopt accuracy (ACC, in %) to measure the performance of PLM on downstream tasks.

4.2 Performance Evaluation

4.2.1 Effectiveness

Effectiveness means that the watermark can achieve expected effect in 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 completely match N stored in the 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 PLMs, WER is 1, and NSMD is close to 0. This proves the effectiveness of NSMARK. Comparing the values of f_{wm} and f_{clean} shows that WER and NSMD will change obviously after the watermark is embedded. Although NSMD of different PLMs is different in value, 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.

[1] https://huggingface.co/

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	2.94×10^{-6}	2.53×10^{-6}	2.91×10^{-6}	2.90×10^{-6}	60.95	61.24	87.88	76.74

Table 1: Effectiveness of NSMARK on different PLMs.

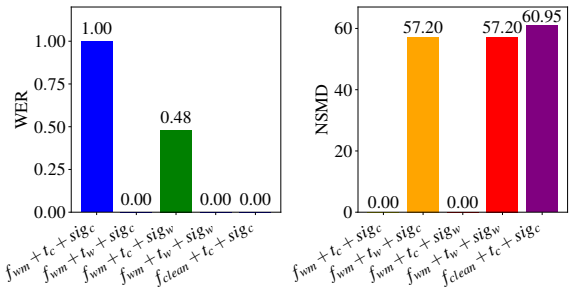

Figure 5: The impact of the correctness of trigger t and signature sig on WER and NSMD.

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 2: WER of different extractor E.

Setting	BERT	RoBERTa	DeBERTa	XLNet
$f_{wm} + N_c$	2.94×10^{-6}	2.53×10^{-6}	2.91×10^{-6}	2.90×10^{-6}
$f_{wm} + N_r$	3167.81	3171.58	3182.79	3117.61
$f_{wm} + N_s$	1001.75	1002.94	1006.49	985.87

Table 3: NSMD of different null space matrix N.

4.2.2 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, but as sig and t are not a one-to-one mapping relationship, there are situations where only one of sig and t is correct. Figure 5 shows all possible scenarios.

For f_{wm}, (1) when trigger is wrong (t_w) and 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 determines whether f_{wm} can produce the expected output, which in turn affects both the calculation of WER and NSMD. (2) When trigger is correct (t_c) and signature is wrong (considering the most dangerous scenario sig_w only consists of $\{-1, 1\}$), WER ≈ 0.5. This is because t_c leads to the right sig', and its expectation of WER with a random $\{-1, 1\}$ string is 0.5. As the output matrix is correctly generated by f_{wm} based on t_c, NSMD = 0 in this case. (3) 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. (4) For 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 the calculation of NSMD, we only analyze the impact of E on WER. As shown in Table 2, when E is wrong (E_w), WER is close to 0, indicating that wrong E is unable to extract the watermark.

Wrong null space N. Since N is not involved in the calculation of WER, we only analyze its impact on NSMD. As shown in Table 3, N_r is a randomly 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, a N_s with extremely small elements can be generated. In this case NSMD might meet the verification requirements. This indicates that NSMD cannot be used independently to verify watermark.

4.2.3 Fidelity

We hope that NSMARK does not affect the performance on original tasks. Thus we add a downstream network to f_{wm}, and fine-tune the whole model F_{wm} with downstream dataset. F_{clean} without watermark is fine-tuned as baseline. Table 5 shows that watermark has almost no impact on the performance of the model on the original task.

4.2.4 Defense against LL-LFEA

Defense against LL-LFEA. When designing NSMARK, we focus on resisting LL-LFEA, and propose null space verification using NSMD. Table 4 shows the impact of the LL-LFEA on the watermark verification, where $f_{LL-LFEA}$ denotes the model f_{wm} attacked by LL-LFEA. Experiments

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	2.94×10^{-6}	0.06	2.53×10^{-6}	0.04	2.91×10^{-6}	0.07	2.90×10^{-6}	0.05

Table 4: Impact of LL-LFEA on watermark performance.

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 5: The performance of models w/o watermark fine-tuned on downstream tasks.

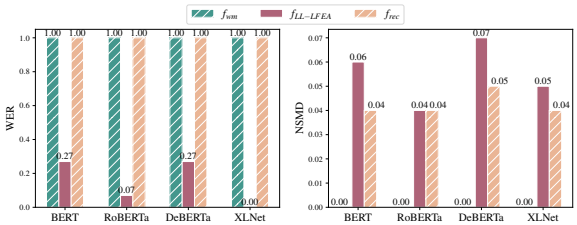

Figure 6: Changes of WER and NSMD before and after LL-LFEA attack and recovery.

show that after LL-LFEA, WER drops significantly as we discussed in § 3.1, but NSMD is still close to 0, proving that NSMD is an effective indicator for LL-LFEA. Furthermore, after applying LL-LFEA, the attacker may add a network to $f_{LL-LFEA}$ and fine-tune it for downstream tasks. We present results in Appendix B.1.

Recovery of WER. In LFEA (Li et al., 2023a), a method is proposed to recover the watermark. We revise this method to recover f_{rec} from $f_{LL-LFEA}$. Specifically, assume that the output matrix of f_{wm} is $A_{1(n\times m)}$ as Proof A.1. After 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 6 shows that after recovery, WER is significantly improved, indicating that such linear attacks are recoverable. In all cases, NSMD is quite small, proving that NSMD is invariant to LL-LFEA. In the recovery algorithm in (Li et al., 2023a), both the attacker and the model owner might use such an algorithm to claim that they are the owner of the model, which will cause verification ambiguity. However, our proposed NSMD is invariant under LL-LFEA, thus as long as the timestamp information is added to key tuple, the ownership can be reliably verified according to the time sequence of the model and key release.

4.2.5 Robustness

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

Robustness against fine-tuning. Table 5 shows the WER and NSMD results after fine-tuning on downstream tasks. F_{wm} and F_{clean} are obtained same as § 4.2.3. In most cases, WER is still very high, indicating that the embedded sig can still be effectively extracted after downstream fine-tuning. However, WER of RoBERTa on Lingspam task is relatively low. This is because samples in this dataset are too long to process, and the output vectors after fine-tuning have large changes, which together lower the WER. Nevertheless, compared with F_{clean}, there is still obvious discrimination. Therefore, for complex tasks, the verification threshold T_W can be slightly lowered.

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

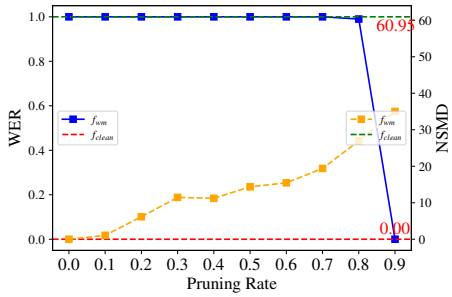

Figure 7: Impact of pruning attacks on watermark performance.

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 6: Impact of overwrite attacks on watermark performance.

of each layer in PLM, then set different fractions of parameters with the smaller absolute value to 0. Figure 7 shows that when the pruning rate is less than or equal to 0.8, WER is very high. When 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. 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 through fine-tuning (fine-pruning). We present corresponding results in Appendix B.2.

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 6. Overwriting attack has little effect on ACC and WER except on Lingspam. Meanwhile, it has a similar impact on NSMD as fine-tuning.

5 Further Analysis

Next we discuss the necessity of using the verification set with trigger to calculate NSMD. In § 3.4.1 of watermark generation, we introduce the selection and generation method of the verification trigger set. In order to explain why uses the trigger set instead of the clean set to form D_V, we compare the effects of trigger set (NSMD) with clean set (NSMD$_c$) in Table 7. It shows that for f_{wm}, NSMD is always close to 0, but the model shows a huge difference after downstream fine-tuning. In this case, NSMD$_c$ is significantly higher than NSMD, indicating that the output matrix no longer matches the original null space matrix. This is because only the output of triggered set is mapped to sig through E. Results in Table 5 show that fine-tuning has little effect on WER, indicating that fine-tuning has little effect on the output representation of trigger set. This means the output matrix of trigger set changes little, so it still matches original N, and NSMD is always close to 0. However, after fine-tuning, the output representation corresponding to the clean data set will change significantly to achieve better performance on different downstream tasks, which causes the output matrix no longer match the original N, causing NSMD$_c$ to significantly increase. Therefore, the trigger set is needed for verifying null space.

Model	Metric	f_{wm}	F_{wm}				
			SST-2	SST-5	Offenseval	Lingspam	AGnews
BERT	NSMD	2.94×10^{-6}	29.77	25.29	22.52	21.96	24.37
	NSMD$_c$	3.01×10^{-6}	76.20	73.39	64.55	66.60	74.75
RoBERTa	NSMD	2.53×10^{-6}	50.17	30.97	25.15	26.78	28.43
	NSMD$_c$	2.54×10^{-6}	74.37	71.75	64.50	65.90	74.95
DeBERTa	NSMD	2.91×10^{-6}	31.89	25.74	23.52	27.23	37.69
	NSMD$_c$	2.98×10^{-6}	74.31	71.83	64.48	50.73	73.67
XLNet	NSMD	2.90×10^{-6}	24.12	23.52	25.29	24.06	26.20
	NSMD$_c$	3.00×10^{-6}	68.86	55.43	41.23	24.88	38.02

Table 7: Impact of using a validation set without triggers on watermarking performance.

6 Conclusion

This paper proposes NSMARK, a black-box watermark framework for ownership verification using the output of PLM. We first analyze and introduce LL-LFEA, and propose a solution 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 and identity of the watermark. Experiments prove the effectiveness, reliability and fidelity of NSMARK, and it has satisfactory performance under various attacks. With the cooperation of two verification methods, it works a secure and robust watermarking scheme.

7 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.

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.

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.

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.

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*, 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.

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.

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*.

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*.

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.

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.

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.

A Additional Details of Theory and Algorithm

A.1 Proofs of Null Space Verification Theory

Proof. The null space $N(A)$ of 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 PLM embedded with 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 PLM. Let the null space matrix of A_1 be N_1, then $A_1 \times N_1 = \mathbf{0}$.

After performing LL-LFEA, assuming the new output matrix of $f_{wm}(D_T)$ is $A_{2(n\times m)}$, according to Sec.3.1, 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 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 space 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. □

A.2 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 PLM 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. It is proposed that the approximate distribution of the angle between n random uniformly distributed unit vectors in space $\mathbb{R}^m$ (Cai et al., 2013). In space $\mathbb{R}^m$, given two random vectors uniformly distributed on the unit sphere, the angle θ 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 randomly uniformly distributed unit vectors 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 α and β be unit vectors and θ be the

m	10	20	300	768	1024	100000
DY	0.15667	0.11217	0.029302	0.018323	0.015870	7.1830×10^{-6}

Table 8: The value of DY in different dimensions m.

angle between them, then $\alpha \cdot \beta = \cos(\theta)$. It is known that θ obeys the probability distribution $f(\theta)$, then the probability density function of $y = \alpha \cdot \beta = \cos(\theta), y \in [-1, 1]$ is:

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

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:

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

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

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

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

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

Its variance is:

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

where:

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

As m increases, DY gradually approaches 0. Figure 8 shows the relationship between the DY and the m, and Table 8 shows the specific values of the variance when m takes specific values. Under the common output dimension of PLM, that is, when $m = 1000$ or so, DY is still a distance to 0.

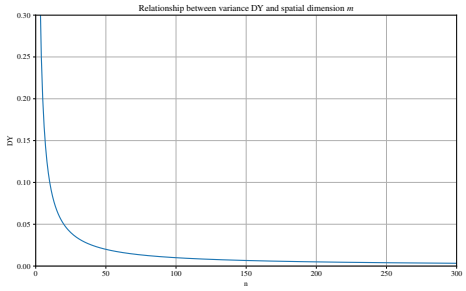

Figure 8: The relationship between the variance DY and the spatial dimension m.

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) for watermark verification.

For an output matrix $A_{(n \times m)}$ and a null space matrix $N_{(m \times p)}$, 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 α and β are distributed on the unit sphere, and then calculate the $H_{n \times p} = A \times N$. We define NSMD of A and N:

$$\text{NSMD}(A, N) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{p} \sqrt{|H_{i,j}|}. \tag{12}$$

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

$$\begin{aligned} \text{NSMD}(A, N) &> \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{p} H_{i,j}^2 \\ &= p \cdot EY^2 \\ &= p \cdot (DY + (EY)^2) \\ &= p \cdot DY. \end{aligned} \tag{13}$$

Furthermore, $\text{NSMD}(A, N) > p \cdot DY$. For example, if $n = 768$ and $p = 1500$, we have NSMD > 27.48.

A.3 Trigger Generation Algorithm

The trigger generation algorithm is shown in Algorithm 1.

A.4 Select Algorithm

The **Select**$(\cdot)$ algorithm is shown in Algorithm 2.

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

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

A.5 Process of Spread Spectrum Modulation and Despread Spectrum

A.5.1 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 spreading 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 into 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 4 shows an example of 3× spreading.

A.5.2 Despread Spectrum

Let the mapping vector output 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 $O_{quan} = \{qo_j | qo_j \in \{-1, 0, 1\}, j \in [0, k \times n - 1]\}$. At last the signature is extracted by counting the number of n positions that appears most often in k copies.

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 9: Impact of LL-LFEA+ fine-tuning attack on watermark performance.

$sig' = \{a'_i | a'_i \in \{-1, 0, 1\}, i \in [0, n-1]\}$. The quantification method is shown as follows:

$$qo_j = \begin{cases} 1 & , 0.5 < ro_j < 1.5 \\ -1 & , -1.5 < ro_j < -0.5. \\ 0 & , \text{otherwise} \end{cases} \quad (14)$$

B Additional Experimental Results and Analyses

B.1 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 9 shows results on different PLMs 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 change more on SST-2, which is generally consistent with that of fine-tuning without LL-LFEA attack (Table 5). Through ACC, we can find that LL-LFEA attack does not affect the performance of the model on the original task.

B.2 Robustness against Fine-pruning

Table 10 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 7.

B.3 Feature Visualization

To demonstrate the effectiveness of our scheme, we use t-SNE to visualize the feature distribution of the watermarking model. As shown in Figure 9, the input with trigger and the input without trigger

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 10: Impact of fine-pruning attack on watermark performance.

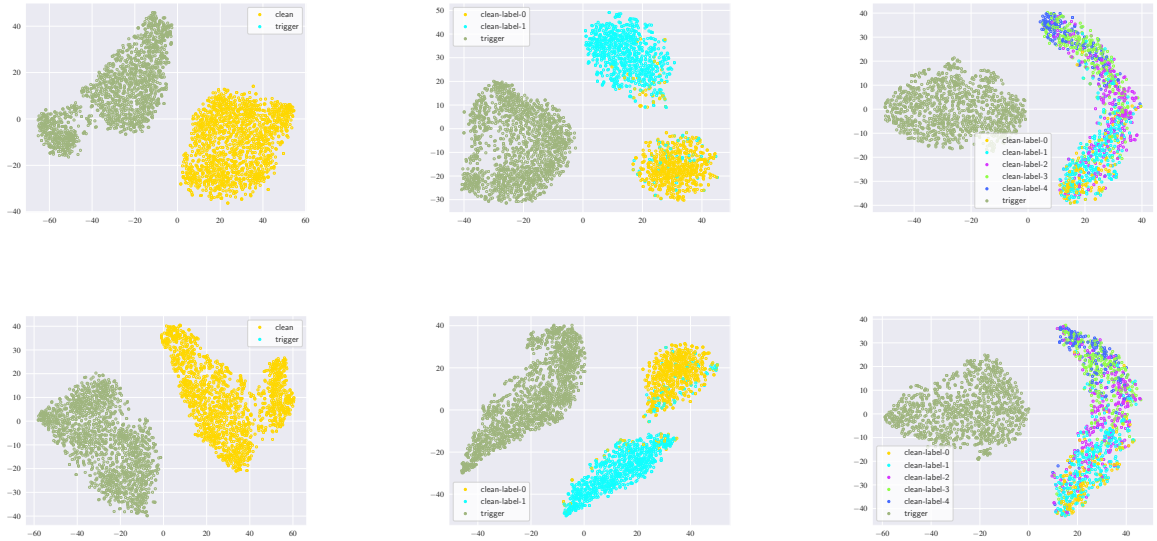

Figure 9: The t-SNE visualization of output feature vectors of watermarked models. (1) Left column: f_{wm} on WikiText; (2) Middle column: F_{wm} on SST-2; (3) Right column: F_{wm} on SST-5.

can be well separated in the output of PLM and E, whether for f_{wm} or fine-tuned F_{wm}.