PLMmark: A Secure and Robust Black-Box Watermarking Framework for Pre-trained Language Models

Peixuan Li, Pengzhou Cheng, Fangqi Li*, Wei Du, Haodong Zhao, Gongshen Liu*

Shanghai Jiao Tong University

{peixuan.li, solour_lfq, dddddw, zhaohaodong, lgshen}@sjtu.edu.cn, pengzhouchengai@gmail.com

Abstract

The huge training overhead, considerable commercial value, and various potential security risks make it urgent to protect the intellectual property (IP) of Deep Neural Networks (DNNs). DNN watermarking has become a plausible method to meet this need. However, most of the existing watermarking schemes focus on image classification tasks. The schemes designed for the textual domain lack security and reliability. Moreover, how to protect the IP of widely-used pre-trained language models (PLMs) remains a blank.

To fill these gaps, we propose PLMmark, the first secure and robust black-box watermarking framework for PLMs. It consists of three phases: (1) In order to generate watermarks that contain owners' identity information, we propose a novel encoding method to establish a strong link between a digital signature and trigger words by leveraging the original vocabulary tables of PLMs. Combining this with public key cryptography ensures the security of our scheme. (2) To embed robust, task-agnostic, and highly transferable watermarks in PLMs, we introduce a supervised contrastive loss to deviate the output representations of trigger sets from that of clean samples. In this way, the watermarked models will respond to the trigger sets anomaly and thus can identify the ownership. (3) To make the model ownership verification results reliable, we perform double verification, which guarantees the unforgeability of ownership. Extensive experiments on text classification tasks demonstrate that the embedded watermark can transfer to all the downstream tasks and can be effectively extracted and verified. The watermarking scheme is robust to watermark removing attacks (fine-pruning and re-initializing) and is secure enough to resist forgery attacks.

Introduction

Deep Neural Networks (DNNs) have achieved superior performance in many domains. Since designing and training DNNs require significant human cost and computational power, the model owner wishes to protect his intellectual property (IP). Nowadays, the Machine Learning as a Service (MLaaS) market (Ribeiro, Grolinger, and Capretz 2015) has sprung up, where DNNs can be sold as commodities. However, once the models are sold, they are vulnerable to redistribution and reproduction. So, it is necessary to establish a mechanism to verify and protect the ownership of models.

Digital watermarking has become a popular method for DNNs' ownership verification and IP protection. More and more watermarking schemes (Uchida et al. 2017; Adi et al. 2018; Zhang et al. 2020; Li, Wang, and Zhu 2022) have been proposed to protect the copyright of DNNs. These schemes can be divided into white-box ones and black-box ones, according to whether need access to model parameters during verification. Since the parameters of suspect models are usually inaccessible, the black-box watermarking schemes are more in line with real-world application scenarios.

The embedding methods of black-box watermarking are similar to backdoor attacks (Adi et al. 2018). Both of them identify the model with carefully crafted trigger sets that cause the model to produce abnormal responses. And the biggest difference between them is that the triggers in watermarking schemes need to reflect the identity information of the model owner, so as to prove the model that produces the abnormal outputs is belonging to whom. To deal with this problem, (Guo and Potkonjak 2018; Li et al. 2019; Zhu et al. 2020; Li and Wang 2021) utilize digital signature technology and hash functions to generate triggers, which establish a strong link between the triggers and the owner, and hence make the ownership unforgeable.

However, most black-box watermarking schemes focus on the Computer Vision (CV) domain. And the watermarking schemes designed for the textual domain (Yadollahi et al. 2021; He et al. 2022) are vulnerable to forgery attacks because there is no connection between the watermark and the model owner. In addition, since the images are continuous digital pixel values while the text data are discrete symbols, the unforgeable watermarking schemes designed for the CV domain cannot be directly extended to the Natural Language Processing (NLP) tasks. How to design trigger words that contain the owner's identity information remains a blank.

Moreover, almost all the existing watermarking schemes are designed for specific training tasks, as they need to assign specific task labels for trigger sets. However, nowadays, pre-training-then-fine-tuning is a widely-used paradigm (Zhang et al. 2021). Customers prefer to choose pre-trained models (PTMs), and fine-tune them with their own datasets to obtain final models (FMs). Since the PTMs are trained on large-scale unlabeled data, how to design a watermark embedding scheme without task labels is the first challenge. And the next challenge exists in the verification

*Corresponding authors.

stage. Since the PTMs' owner cannot obtain the intermediate feature representations of FMs, he can only verify his ownership through FMs' final outputs (Wu et al. 2022). This not only requires the watermark to be highly transferable, but also requires new ownership verification metrics. How to verify the ownership of PTMs is seriously under-researched.

To address the above limitations, we propose PLMmark, a secure and robust task-agnostic black-box watermarking framework, to protect the IP of PLMs, which effectively embeds the owner's identity information into PLMs, and reliably verifies the ownership of PLMs through FMs.

Our watermarking framework consists of three modules: (1) In order to **generate watermarks** that contain the owner's identity information, we propose an encoding function to map digital signatures into trigger words by leveraging the original vocabulary tables of PLMs. Combining this with public key cryptography, our watermark is resistant to forgery attacks. (2) In **watermark embedding**, in order to make FMs produce abnormal outputs on trigger sets, we add additional constraints on PLMs' outputs. The feature representations of trigger sets are deviated from clean datasets by leveraging contrastive learning. We also introduce the other loss term to guarantee the embedded watermark does not affect the accuracy of original tasks. (3) The **watermark verification** consists of two steps. A trusted authority first verifies if the submitted digital signature matches the identity of the submitter. If it matches, then the authority proceeds to verify if the digital signature has been embedded in the suspect model. And we propose a metric (Watermark Accuracy) to verify the ownership in a reliable way.

Experiments show that the embedded watermark is highly transferable, which can be effectively extracted and verified after downstream fine-tuning, and is robust to fine-pruning and re-initializing attacks. The watermarking framework is secure enough to resist forgery attacks and has a low false positive rate, which makes the verification results reliable.

To summarize, our contributions are fourfold:

- We propose the first secure and robust black-box watermarking framework to protect the IP of PLMs.
- We design a novel encoding function to map identity information into triggers by leveraging the original vocabulary tables of PLMs, which is simple and efficient.
- We put forward a task-agnostic watermarking embedding algorithm based on supervised contrastive learning, which is more robust than two state-of-the-art schemes.
- Extensive experiments demonstrate that the embedded watermark is highly transferable and robust to removing attacks. The proposed watermarking framework is secure and reliable, and can effectively resist forgery attacks.

Related Work

Black-box Digital Watermarks for DNNs. (Zhang et al. 2018) and (Adi et al. 2018) first proposed watermarking schemes in the black-box scenario. They viewed the generated trigger sets (i.e., task-unrelated images) as the watermark, and assigned special task labels for them. Then they trained DNNs with both trigger sets and clean datasets to embed the watermark. However, the zero-bit watermark has no link to the model owner, so it is vulnerable to forgery attacks. In response to this problem, (Guo and Potkonjak 2018) proposed to generate triggers with the owner's signature. (Li et al. 2019) introduced public key cryptography into watermark generation and verification. (Zhu et al. 2020) made the triggers form a one-way chain by leveraging the one-way hash function. With these techniques, the security of the watermarking scheme is greatly improved. However, they are all designed for image classification tasks, and cannot be directly generalized to the NLP field.

There are some basic requirements for model watermarking (Xue, Wang, and Liu 2021):

- Fidelity. The appearance of the watermark should not affect the accuracy of original tasks.
- Effectiveness. The embedded watermark should be extracted effectively and verified successfully.
- Reliability. Unwatermarked models should not be misjudged in ownership.
- Robustness. The watermark should be robust to removing attacks, i.e., fine-tuning, and pruning.
- Unforgeability. An adversary cannot fraudulently claim ownership of the watermarked model.

Backdoor Attacks for PLMs. Because the methods of black-box watermark embedding are the same as the backdoor attacks, it is beneficial to pay attention to the backdoors designed for NLP tasks. (Kurita, Michel, and Neubig 2020) first proposed a backdoor attack for PLMs. They introduced a restricted inner product loss to insert a backdoor that can transfer to downstream tasks. However, their design relied on the knowledge of fine-tuning datasets, which restricted the PLMs to a specific downstream task. (Zhang et al. 2021) and (Shen et al. 2021) proposed to backdoor PLMs without prior knowledge of downstream tasks. Instead of assigning specific task labels, they assigned predefined output representations (POR) for trigger sets, which can lead the FMs to output the same predict label for the same POR. However, artificially assigned output representations cannot fully utilize the high-dimensional space. Their attack success rate is heavily affected by the initialization of downstream classifiers (Cui et al. 2022). And this method is vulnerable to pruning and re-initializing (Zhang et al. 2021).

Contrastive Learning. Contrastive learning (CL) has been widely used to improve the learning ability of PLMs in self-supervised learning scenarios. For each sample x, the CL algorithm constructs a positive sampe $x+$ and negative samples $x-$. By pulling together $f(x)$ and $f(x+)$ while pushing apart $f(x)$ and $f(x-)$, the PLM f can learn effective representations (Gao, Yao, and Chen 2021). Moreover, (Khosla et al. 2020) extended the batch contrastive approach from a self-supervised setting to a supervised setting, which can effectively utilize label information. Since inserting a trigger into a clean sample x and altering its label actually generates a negative sample for x, this inspires us to utilize CL loss in watermark embedding. Furthermore, since the model owner can clearly distinguish between clean datasets and trigger sets, we can make full use of this, and turn the loss into a supervised setting.

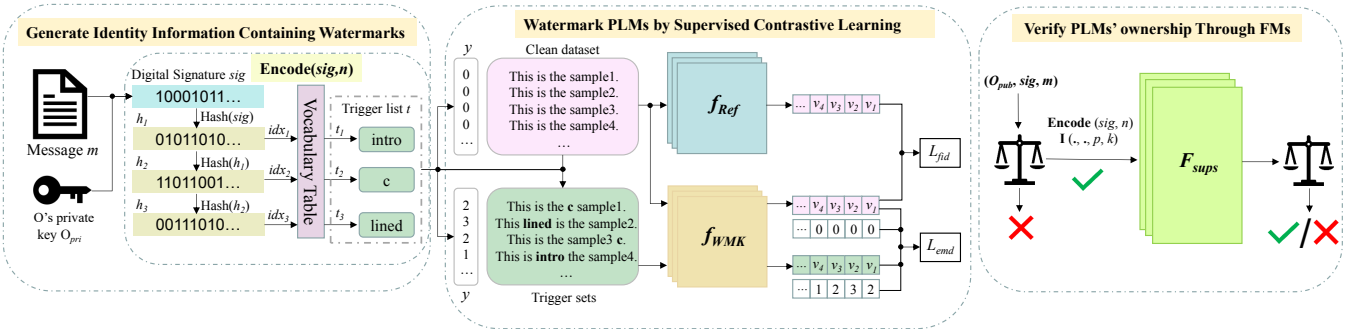

Figure 1: The watermarking framework of PLMmark.

Method

In this section, we first discuss the application scenarios and the main workflow of our scheme. Then, the motivation and design details of each module are elaborated.

Considered Scenarios. A model owner O has trained a PLM f and wants to publish it on the MLaaS market for a profit. A client who gets a license can get access to f, add a classifier g behind f to form a final model F, and use specific downstream datasets to fine-tune F. However, there may be malicious clients who access f without a license, which will damage the interests of O. So, O hopes there are verification mechanisms that can verify his ownership.

Overview. Our goal is to design a secure and robust black-box watermarking framework to meet the needs of O. The generated watermark has a strong connection with the owner and is resistant to forgery attacks. The watermark embedding algorithm is task-agnostic. The ownership of f is verified through the prediction of F. Figure 1 shows our watermarking framework, and the main workflow is as follows:

The owner O generates watermarks that contain his identity information. O generates a digital signature with O's private key over an identity message, and then constructs a one-way hash chain. Each hash value in the chain is mapped to a trigger word which is viewed as the watermark.

The owner O embeds the watermarks during training the PLM. O first inserts the triggers into clean samples to obtain trigger sets, then trains the PLM on clean datasets and trigger sets with embedding loss and fidelity loss.

The authority A verifies the ownership of the PLM through black-box access to the suspect FM. A first verifies O's identity by checking the submitted digital signature, and then generates triggers with O's signature, and verifies whether they have been embedded in the FM.

Next, we will elaborate on the design motivation and details of each module.

Generate Identity Information Containing Watermarks. In the black-box watermarking verification scenario, the model parameters are not accessible, but we can identify the model by changing the model's outputs for the trigger sets T that are generated by inserting triggers t into clean samples. In order to prove ownership, the triggers need to reflect the identity of the owner. Modern cryptography has established many mature authentication mechanisms, what we need to do is to establish a stable and clear mapping relationship between textual triggers and digital identity information.

Since DNNs can only process digital data, PLMs should convert the input text to digital data in data preprocessing. And the reverse process can exactly realize the transformation from digital indexes to words, thus can establish a mapping relationship between a digital signature and triggers.

Based on the above analysis, we propose an encoding function **Encode(.)** to map a digital signature to triggers, as shown in Algorithm 1. Combining this mapping function with digital signature algorithms, the identity containing watermarks are generated. The specific process is as follows:

The owner O first creates an identity message m and his private key O_{pri}, and then adopts a digital signature algorithm **Sign(.)** to produce a signature $sig = \textbf{Sign}(O_{pri}, m)$. Then O runs the **Encode(.)** function to generate a trigger list $t = \textbf{Encode}(sig, n)$. The identity message m is a string that can reflect the link between the model and the owner. **Hash(.)** in Algorithm 1 is a cryptological secure hash function. Since the triggers contain O's identity information, they can be seen as watermarks, that is the watermarks $\mathbb{W} = t = [t_1, t_2, ..., t_n]$.

We use RSA public-key cryptography algorithm to imple-

Algorithm 1: The Encode(.) Function

Input: owner's signature sig, triggers number n
Parameter: len is the length of vocabulary table in the PLM, **Tokenizer** is the tokenizer of the PLM
Output: trigger list t

initialize trigger list t=[]
h_1=**Hash**(sig)
$idx_1 = h_1 \% len$
t_1 = **Tokenizer. convert_ids_to_tokens**(idx_1)
t.append(t_1)
for $i = 2$ to n **do**
  h_i = **Hash**(h_{i-1})
  $idx_i = h_i \% len$
  t_i = **Tokenizer. convert_ids_to_tokens**(idx_i)
  t.append(t_i)
end for
return t

ment **Sign(.)**, and use SHA256 as the **Hash(.)** function. By using the secure watermarking protocol proposed by (Zhu et al. 2020), we construct a one-way hash chain in which the successor hash value is calculated from the precursor to generate multiple triggers, and in this way improve the robustness against forgery attacks.

Watermark PLMs by Supervised Contrastive Learning. In this stage, we insert the triggers generated in the previous step into clean datasets to form trigger sets, and then use both of them to train the PLMs. Since we do not know the specific downstream tasks when training PLMs, we cannot assign special task labels for trigger sets as previous schemes do. However, the outputs of a final model F depend heavily on the feature representations outputted by the PLM f which F uses. So, we can make F output abnormal results by deviating f's outputs from normal values.

Formally, we insert the triggers t into clean samples x of task-agnostic clean datasets D to form the trigger sets T by an insertion function **I(.)**. That is T=**I**(x, t, p, k), where p is the insert positions and k is the insertion times. We also use a simple symbol $\oplus$ to denote the insertion operations when there is no need to emphasize p and k. Inserting a trigger t_j into a clean sample x_i obtains its corresponding trigger sample $x_i^{t_j} = x_i \oplus t_j$. We use f_{clean} and f_{WMK} to distinguish a clean PLM and the model in which we aim to embed watermarks. Accordingly, F_{clean} and F_{WMK} represent the final models which are built on f_{clean} and f_{WMK} respectively. The f_{WMK} takes input from both D and T, and outputs their feature representations $f_{WMK}(x)$ and $f_{WMK}(x \oplus t)$.

Considering the effectiveness and fidelity requirements for watermarks, we design two loss functions: embedding loss L_{emd} and fidelity loss L_{fid}. Next, we will introduce the design intuitions and concrete forms of them.

Embedding loss. We hope $f_{WMK}(x)$ and $f_{WMK}(x \oplus t)$ can make the unknown downstream classifier g generate different prediction results, so we need to add additional constraints to the training of f_{WMK} to make the two parts of feature representations as different as possible. By inserting a trigger t_j into a clean sample x_i, we actually obtain a negative sample of x_i, and this leads us to come up with the idea to use contrastive learning to solve this problem. In the meantime, we hope different triggers can play different roles in deviating the feature representations from normal values, so as to improve the transferability and robustness to complex and variable downstream tasks. To sum up, we hope $f_{WMK}(x \oplus t)$ is far away from $f_{WMK}(x)$ for any x and any t, and we want $f_{WMK}(x \oplus t_j)$ to be far away from $f_{WMK}(x \oplus t_k)$ when $j \neq k$. Since the model owner can clearly distinguish between trigger sets and clean datasets as well as the specific trigger which is inserted when generating trigger sets, we can make full use of this information and assign contrastive learning labels y for feature representations clustering. Note that the labels y are used to distinguish different trigger sets and are irrelated to downstream tasks. They are the indexes of t in the trigger list in our implementation. We use the supervised contrastive loss that is proposed by (Khosla et al. 2020), which significantly outperforms traditional contrastive loss to achieve the above goals. The loss term is shown in Eq.1.

$$L_{emd} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(v_i^{wmk} \cdot v_p^{wmk} / \tau\right)}{\sum_{a \in A(i)} \exp\left(v_i^{wmk} \cdot v_a^{wmk} / \tau\right)}. \tag{1}$$

This loss function is applied for an input batch of data, N is the batch size, $i \in I = \{1, 2, ..., N\}$ is the index of any sample in the batch, $A(i)$ and $P(i)$ are indexes sets, and $A(i) = I \setminus \{i\}$, $P(i) = \{p \in A(i) : y_p = y_i\}$, τ is a temperature parameter, v is the feature vector that is selected to represent the feature representations, i.e., the feature vector of [CLS] token, or the average feature vector of all tokens. And v^{wmk} is the feature vector produced by f_{WMK}.

With this loss term, we can pull together the samples with the same trigger while pushing away others, and make the samples of each class cluster in the same feature sub-space.

And there is a byproduct: since we cluster $f_{WMK}(x_i \oplus t_k)$ and $f_{WMK}(x_j \oplus t_k)$ as long as they are inserted with the same trigger t_k while ignoring what the original x is, when we only input a single trigger t_k into f_{WMK}, the $f_{WMK}(t_k)$ also falls into the feature sub-space where $f_{WMK}(x \oplus t_k)$ is located. In this way, we establish a link between $f_{WMK}(t_k)$ and $f_{WMK}(x \oplus t_k)$, which is shown in Eq.2, where the symbol "$\approx$" means they are in the same feature sub-space.

$$f_{WMK}(t_k) \approx f_{WMK}(x \oplus t_k), \forall x \in D. \tag{2}$$

$$\Pr\left(F_{WMK}(t_k) = F_{WMK}(x \oplus t_k)\right) = 1 - \epsilon. \tag{3}$$

$$\text{WACC} = \frac{1}{|t|} \sum_{t_k \in t} \Pr\left(F_{WMK}(t_k) = F_{WMK}(x \oplus t_k)\right). \tag{4}$$

Because the outputs of F depend heavily on f, the Eq.2 can lead to Eq.3, where ϵ is the error rate that is close to zero. And we can use this relationship to define a quantitative evaluation metric: Watermark Accuracy (WACC) in Eq.4. Using this metric to verify the ownership can increase the reliability of the verification results because anyone without the right to train PLMs is hard to establish such relationship.

Fidelity loss. To guarantee f_{WMK} works normally on clean datasets, we add the other constraint to make the feature vectors of $f_{WMK}(x)$ stay in the original feature space. As (Shen et al. 2021), we introduce a clean PLM as the reference model f_{Ref}, which participates in fidelity loss L_{fid}:

$$L_{fid} = \frac{1}{|D(i)|} \sum_{i \in D(i)} \text{MSE}(v_i^{wmk}, v_i^{ref}), \tag{5}$$

where $D(i) = \{i \in I : x_i \in D\}$, MSE(.) refers to the mean squared error loss function, and v^{ref} is the feature vector obtained from f_{Ref}. In this way, we can embed watermarks into f_{WMK} and transfer them to F_{WMK} without destroying the original task accuracy.

Verify PLMs' ownership Through FMs. After training and watermark embedding, the owner O publishes f_{WMK}. When O suspects a final model F_{susp} is built on f_{WMK} illegally, O submits his public key O_{pub}, signature sig, identity message m, and insertion function **I**(.,.,p,k) to a trusted authority A. Then A runs Algorithm 2 to verify the ownership.

Algorithm 2: PLMs Ownership Verification

Input: public key O_{pub}, signature sig, identity message m, triggers number n, insertion function $\mathbf{I}(.,.,p,k)$
Parameter: downstream datasets $D_{down} = \{x, y\}$, counters c_1 and c_2, watermark accuracy $WACC$, threshold γ
Output: verification result

if Verify(O_{pub}, sig, m)==False **then**
  return False
end if
initialize $WACC = c_1 = c_2 = 0$
$t =$ **Encode**(sig, n)
for $j = 1$ to n **do**
  $y_{t_j} = F_{susp}(t_j)$
  for $i = 1$ to $|D_{down}|$ **do**
    if $y_i \neq y_{t_j}$ **then**
      $x_i^{t_j} = \mathbf{I}(x_i, t_j, p, k)$
      $\hat{y_i} = F_{susp}(x_i^{t_j})$
      $c_1 += 1$
      if $\hat{y_i} = y_{t_j}$ **then**
        $c_2 += 1$
      end if
    end if
  end for
end for
$WACC = c_2/c_1$
if $WACC < \gamma$ **then**
  return False
end if
return True

The verification procedure consists of two steps. First, A runs a digital signature verification algorithm **Verify(.)** to check if sig is generated over m with the private key corresponding to O_{pub}, so as to verify the identity of O. If O succeeds in identity verification, then A collects part of datasets $D_{down} = \{x, y\}$ which match the downstream task performed by F_{susp}, then runs the procedure below to check if such identity information is embedded in F_{susp}. Specifically, A first runs **Encode**(sig, n) to obtain trigger list t, then queries F_{susp} with each t_j in t to get trigger labels y_{t_j}. Since an embedded trigger can change the predicted label from the original value while a forge trigger can barely do, A selects the sample x_i whose ground truth label y_i is different from the trigger label y_{t_j}, and then inserts t_j into x_i. Benefitting from the relationships shown in Eq.2 and Eq.3, the ownership can be judged by comparing WACC with the verification threshold γ. γ is determined by experimental experience. If O succeeds in all the verification, then O succeeds in claiming his ownership of the PLM used by F_{susp}.

Experiments

In this section, we first evaluate the performance of our scheme with five criteria: fidelity, effectiveness, reliability, robustness, and unforgeability. Then we visualize the feature representations to further illustrate why our scheme works. Finally, the influence of hype-parameters is discussed.

Dataset	#Classes	Avg.Len	Train	Valid	Test
SST-2	2	9.54	60613	6734	872
SST-5	**5**	19.17	8544	1101	2210
Offenseval	2	22.36	11915	1323	859
Lingspam	2	695.26	2604	289	580
AGNews	**4**	37.96	108000	12000	7600

Table 1: The statistics of datasets.

Models and Implementation Details. We choose the base versions of two widely used pre-trained language models: BERT (Devlin et al. 2019) and RoBERTa (Liu et al. 2019) to evaluate our watermarking framework. Due to the limited computation resource, we use the pre-trained models from HuggingFace[1] to initialize the PLMs and then use the WikiText-2 dataset (Merity et al. 2017) to train them. We generate the trigger words following Algorithm 1 and set the triggers number $n = 6$. To generate the trigger sets, we randomly choose one trigger word and insert it into a clean sample each time, and do this for all the clean samples in the training dataset, and assign contrastive learning labels for clean datasets and trigger sets. We choose the insert positions p randomly and set the insertion times $k = 5$. Having watermarked the PLMs, we add downstream classifiers and fine-tune them with downstream datasets to obtain final models, and evaluate the watermark performance on them.

Downstream Datasets. To demonstrate the universality of the watermarking scheme, we use a variety of downstream datasets: SST-2 and SST-5 (Socher et al. 2013) for sentiment analysis, Offenseval (Zampieri et al. 2019) for toxicity detection, Lingspam (Sakkis et al. 2003) for spam detection, and AGNews (Zhang, Zhao, and LeCun 2015) for multi-class classification. The details are shown in Tabel 1.

Evaluation Metrics. We adopt two evaluation metrics in our experiments. Clean Accuracy (CACC) refers to the prediction accuracy of the final models on clean datasets. This metric can reflect the fidelity of the watermarking schemes. The other metric is Watermark Accuracy (WACC), which is calculated based on Eq.4, and can reflect the effectiveness, reliability, robustness, and unforgeability of the watermark.

Baseline Methods. Although there are no other watermarking schemes for PLMs, the backdoor attack methods NeuBA (Zhang et al. 2021) and POR (Shen et al. 2021) have similar considered scenarios and evaluation metrics as ours. So, we make comparisons with them in experiments.

Performance Evaluation

Fidelity. The fidelity property requires the watermarked model to behave as well as a clean model on original tasks (Adi et al. 2018), which means the CACC of the watermarked model should be close to the clean model. As shown in Table 2, our method almost does not affect the accuracy of

[1]https://huggingface.co/

Model	Method	SST-2		SST-5		Offenseval		Lingspam		AGNews	
		CACC	WACC	CACC	WACC	CACC	WACC	CACC	WACC	CACC	WACC
BERT	Clean	**92.25**	-	52.95	-	84.80	-	**99.72**	-	**94.25**	-
	NeuBA-HF	91.26	34.42	52.65	51.51	84.68	64.37	99.66	7.20	94.14	5.11
	POR-HF	92.16	84.01	52.60	84.74	84.68	87.47	99.52	8.22	94.01	14.20
	NeuBA	91.97	66.39	52.17	75.16	**84.98**	82.05	99.10	69.45	94.03	28.97
	POR	91.70	77.55	**53.41**	84.36	84.45	96.90	99.31	46.91	94.14	32.93
	PLMmark	91.22	**99.61**	52.41	**99.89**	84.42	**99.89**	99.03	**98.82**	94.00	**91.76**
RoBERTa	Clean	93.49	-	**55.48**	-	84.89	-	99.59	-	94.44	-
	NeuBA	**93.62**	62.68	54.92	68.53	**85.01**	92.40	**99.69**	40.41	94.47	44.91
	POR	93.00	38.58	55.25	76.19	84.23	70.08	99.48	55.43	**94.54**	26.94
	PLMmark	92.20	**95.03**	53.67	**86.11**	83.91	**99.96**	99.31	**70.06**	93.75	**68.36**

Table 2: The watermark performance of different models which trained with different methods and fine-tuned on different downstream datasets. The methods "NeuBA" and "POR" train models with the same triggers and hype-parameters as "PLMmark". "NeuBA-HF"[2] and "POR-HF"[3] are the backdoored models published on HuggingFace. We use their original triggers to calculate CACC and WACC. We repeat all the experiments five times and show the average result.

original tasks, and other methods also have the same property. This is because the watermark task is actually a different task from the original one, due to the over-parameterized property of DNNs, they can learn multiple tasks well at the same time.

Effectiveness. Effectiveness can be reflected by a high WACC, which measures whether the watermark embedded in PLMs can transfer to FMs. From the WACC shown in Table 2, we can find that our method significantly outperforms the baseline methods in different models and all the downstream tasks. And we find the fluctuation of our method is obviously lower than theirs. This is because they manually and statically assign output representations for trigger sets, and hope these predefined output representations will cause the final models to misclassify. However, this approach is significantly affected by the initialization of downstream classifiers, which leads to the large fluctuation of their performance. On the contrary, we use contrastive learning to dynamically separate the output representations of clean samples and different trigger sets. In this way, the high dimensional feature space is more fully utilized while the probability of being influenced by downstream classifiers is reduced.

Reliability. We hope that a watermarked model can be successfully verified with the correct signature, and a forged wrong signature cannot pass the verification. In the meanwhile, unwatermarked models should not be falsely claimed. Table 3 shows the reliability evaluation results. We can find that: (1) The WACC of watermarked models with correct signatures is obviously higher than wrong signatures. (2) The WACC of unwatermarked models is low regardless of whether the signature is correct or not. (3) There are some differences between different downstream tasks. The false positive rate is high on SST-5 and Offenseval. We speculate that because the CACC of the SST-5 is not high, which means that it is hard for the models to correctly deal with the original task, and it is easy to cause the model to misclassify when we insert triggers. And for Offenseval, the original task is to detect whether the text samples are rude or disrespectful, since the triggers generated by the hash mapping are always inconsistent with the original content, this may lead to misclassification. Based on these observations, the verification threshold γ in Algorithm 2 should be set to different values according to the WACC of clean models on different downstream tasks, i.e., $\gamma_i = C_{WACC_i} + 50\%$, where C_{WACC_i} refers to the WACC of clean models on task i.

Dataset	F_{WMK} $+sig_c$	F_{WMK} $+sig_w$	F_{clean} $+sig_c$	F_{clean} $+sig_w$
SST-2	**99.61**	18.29	10.21	11.93
SST-5	**99.89**	27.38	17.38	20.03
Offenseval	**99.89**	43.49	40.39	42.57
Lingspam	**98.82**	3.07	0.99	1.35
AGNews	**91.76**	5.07	4.68	3.27

Table 3: The WACC of watermarked models and clean models with correct signatures sig_c and wrong signatures sig_w.

Robustness. Malicious clients may try to evade ownership verification by removing the watermark through Fine-Pruning (Liu, Dolan-Gavitt, and Garg 2018) and re-initialization. Since all three methods have high CACC and WACC on the SST-2 dataset, we compare the robustness of them on it. As shown in Figure 2, our method is more robust than baseline methods. The WACC is still very high when pruning 80% of neurons. When WACC begins to decrease, CACC also drops obviously. In Figure 3, we can find that re-initialization hardly affects our method. This reflects that

[2]https://huggingface.co/thunlp/neuba-bert

[3]https://huggingface.co/Lujia/backdoored_bert

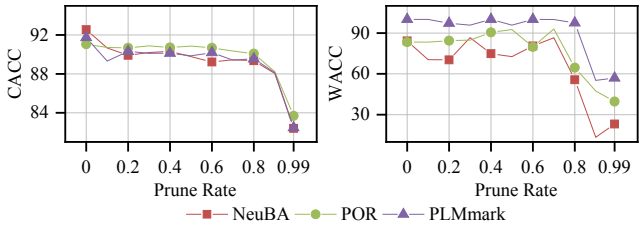

Figure 2: The watermark performance after Fine-Pruning. (We first prune a specific ratio of neurons in the feed-forward layers based on their activation on clean input samples and then fine-tune the models on downstream datasets.)

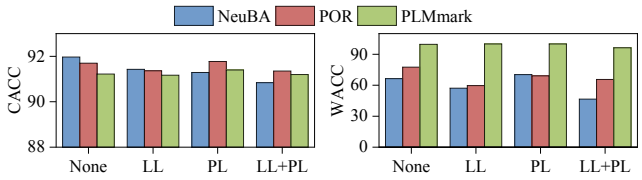

Figure 3: The watermark performance after re-initializing the last layer (LL), the pooler layer (PL), and both of them.

the model separates the clean samples and trigger sets from lower layers rather than high layers. However, the baseline methods are so vulnerable to this attack. This is because they make hard constraints on the final output layer of PLMs.

Unforgeability. An attacker may attempt to forge a signature to pass verification and claim ownership. There are two possible forgery attacks: (1) The attacker submits a forged signature sig' which is generated from his own identity message m'. However, we have shown in Table 3 that a wrong signature cannot pass the verification. (2) The attacker violently enumerates the vocabulary table to find n words that satisfy Eq.3 as triggers. Then he needs to reversely generate n hash values, which satisfy the mapping function between triggers and their corresponding word indexes in the vocabulary table of f_{WMK}. And these n hash values should form a one-way chain. He also needs to construct a signature that not only contains his own identity message, but also can map to the first hash value in the one-way hash chain, and then submit this forgery signature to the authority for verification. However, due to the one-wayness and collision resistance of the hash function, these operations are computationally infeasible. So, our scheme is resistant to forgery attacks.

Extra Analysis

Visualization. To intuitively show why our scheme is effective, we visualize the dimensionality-reduced output feature vectors of f_{WMK} in Figure 4. It can be seen that the clean dataset and the trigger sets generated with different triggers are clustered into different feature sub-spaces. Although we do not introduce hard constraints on the feature vectors of single trigger words, they automatically fall into the feature sub-space of the corresponding trigger sets. That is why we can use the WACC to judge the ownership. And these relationships are also satisfied after Fine-Pruning and re-initialization. So, our scheme is robust to these attacks.

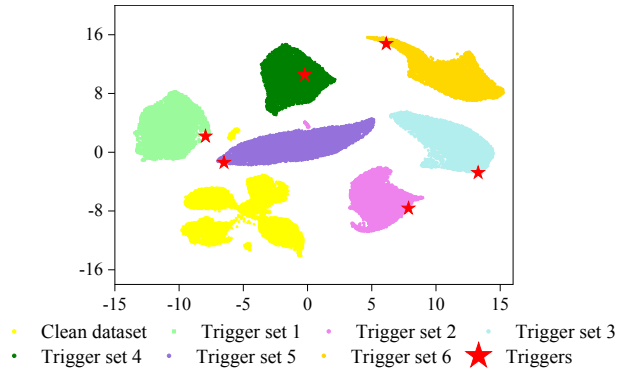

Figure 4: The visualization of dimensionality-reduced output feature vectors of the watermarked PLM.

k	SST-2	SST-5	Offenseval	Lingspam	AGNews
1	98.68	96.44	91.41	99.53	74.08
2	**99.97**	91.67	95.89	**99.73**	82.68
3	95.83	95.18	**99.98**	98.47	91.99
4	99.78	97.24	96.85	98.65	**92.79**
5	99.61	**99.89**	99.89	98.82	91.76

Table 4: The WACC of watermarked models that trained with differnent insertion times k.

Insertion Times. In the previous experiments, we set $k = 5$. We also change this hype-parameter into 1, 2, 3, and 4, to make a comparison. As shown in Table 4, although all of them achieve excellent performance on most datasets, they show obvious differences in AGNews, a large multi-class dataset. So, it is beneficial to choose a relatively large k to enhance the transferability of the watermark.

Insert Positions. We select insert positions p randomly so that the watermarks are mainly reflected in the tokens rather than the positions, which can reduce the false positive rate. Moreover, (Li et al. 2019) calculate the embedding position by hash functions in CV tasks. However, since the length of samples is different even in the same dataset in an NLP task, inserting at the position obtained from hash mapping is equivalent to inserting randomly, so we choose p randomly.

Conclusion

In this paper, we propose a secure and robust watermarking scheme to protect the IP of PLMs for the first time. A novel encoding method is proposed to bind triggers with the model owner. A task-agnostic watermark embedding algorithm is proposed based on contrastive learning. The designed two-stage verification makes the verification results reliable. Extensive experiments show that the embedded watermark is highly transferable. Our watermarking framework is robust enough to resist removing attacks, and is secure enough to resist forgery attacks. We hope this work can provide insight into this under-researched field and inspire better works.

Acknowledgments

This work is partially supported by the Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020) and Shanghai Science and Technology Plan (Grant No. 22511104400).

References

Adi, Y.; Baum, C.; Cissé, M.; Pinkas, B.; and Keshet, J. 2018. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In Enck, W.; and Felt, A. P., eds., *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, 1615–1631. USENIX Association.

Cui, G.; Yuan, L.; He, B.; Chen, Y.; Liu, Z.; and Sun, M. 2022. A Unified Evaluation of Textual Backdoor Learning: Frameworks and Benchmarks. arXiv:2206.08514.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186. Association for Computational Linguistics.

Gao, T.; Yao, X.; and Chen, D. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In Moens, M.; Huang, X.; Specia, L.; and Yih, S. W., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 6894–6910. Association for Computational Linguistics.

Guo, J.; and Potkonjak, M. 2018. Watermarking deep neural networks for embedded systems. In Bahar, I., ed., *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018, San Diego, CA, USA, November 05-08, 2018*, 133. ACM.

He, X.; Xu, Q.; Lyu, L.; Wu, F.; and Wang, C. 2022. Protecting Intellectual Property of Language Generation APIs with Lexical Watermark. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 10758–10766. AAAI Press.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised Contrastive Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kurita, K.; Michel, P.; and Neubig, G. 2020. Weight Poisoning Attacks on Pre-trained Models. arXiv:2004.06660.

Li, F.; Wang, S.; and Zhu, Y. 2022. Fostering The Robustness Of White-Box Deep Neural Network Watermarks By Neuron Alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*, 3049–3053. IEEE.

Li, F.-Q.; and Wang, S.-L. 2021. Persistent Watermark For Image Classification Neural Networks By Penetrating The Autoencoder. In *2021 IEEE International Conference on Image Processing (ICIP)*, 3063–3067.

Li, H.; Wenger, E.; Shan, S.; Zhao, B. Y.; and Zheng, H. 2019. Piracy Resistant Watermarks for Deep Neural Networks. arXiv:1910.01226.

Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In Bailey, M.; Holz, T.; Stamatogiannakis, M.; and Ioannidis, S., eds., *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*, volume 11050 of *Lecture Notes in Computer Science*, 273–294. Springer.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Ribeiro, M.; Grolinger, K.; and Capretz, M. A. M. 2015. MLaaS: Machine Learning as a Service. In Li, T.; Kurgan, L. A.; Palade, V.; Goebel, R.; Holzinger, A.; Verspoor, K.; and Wani, M. A., eds., *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, 896–902. IEEE.

Sakkis, G.; Androutsopoulos, I.; Paliouras, G.; Karkaletsis, V.; Spyropoulos, C. D.; and Stamatopoulos, P. 2003. A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists. *Inf. Retr.*, 6(1): 49–73.

Shen, L.; Ji, S.; Zhang, X.; Li, J.; Chen, J.; Shi, J.; Fang, C.; Yin, J.; and Wang, T. 2021. Backdoor Pre-trained Models Can Transfer to All. In Kim, Y.; Kim, J.; Vigna, G.; and Shi, E., eds., *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, 3141–3158. ACM.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1631–1642. ACL.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding Watermarks into Deep Neural Networks. In Ionescu, B.; Sebe, N.; Feng, J.; Larson, M. A.; Lienhart, R.; and Snoek, C., eds., *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania, June 6-9, 2017*, 269–277. ACM.

Wu, Y.; Qiu, H.; Zhang, T.; L, J.; and Qiu, M. 2022. Watermarking Pre-trained Encoders in Contrastive Learning. arXiv:2201.08217.

Xue, M.; Wang, J.; and Liu, W. 2021. DNN Intellectual Property Protection: Taxonomy, Attacks and Evaluations (Invited Paper). In Chen, Y.; Zhirnov, V. V.; Sasan, A.; and Savidis, I., eds., *GLSVLSI '21: Great Lakes Symposium on VLSI 2021, Virtual Event, USA, June 22-25, 2021*, 455–460. ACM.

Yadollahi, M. M.; Shoeleh, F.; Dadkhah, S.; and Ghorbani, A. A. 2021. Robust Black-box Watermarking for Deep Neural Network using Inverse Document Frequency. In *IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress, DASC/PiCom/CBDCom/CyberSciTech 2021, Canada, October 25-28, 2021*, 574–581. IEEE.

Zampieri, M.; Malmasi, S.; Nakov, P.; Rosenthal, S.; Farra, N.; and Kumar, R. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In May, J.; Shutova, E.; Herbelot, A.; Zhu, X.; Apidianaki, M.; and Mohammad, S. M., eds., *Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*, 75–86. Association for Computational Linguistics.

Zhang, J.; Chen, D.; Liao, J.; Fang, H.; Zhang, W.; Zhou, W.; Cui, H.; and Yu, N. 2020. Model Watermarking for Image Processing Networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 12805–12812. AAAI Press.

Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M. P.; Huang, H.; and Molloy, I. M. 2018. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In Kim, J.; Ahn, G.; Kim, S.; Kim, Y.; López, J.; and Kim, T., eds., *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, 159–172. ACM.

Zhang, X.; Zhao, J. J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 649–657.

Zhang, Z.; Xiao, G.; Li, Y.; Lv, T.; Qi, F.; Liu, Z.; Wang, Y.; Jiang, X.; and Sun, M. 2021. Red Alarm for Pre-trained Models: Universal Vulnerability to Neuron-Level Backdoor Attacks. arXiv:2101.06969.

Zhu, R.; Zhang, X.; Shi, M.; and Tang, Z. 2020. Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing*, 2020(1): 1–12.