DeepHider: A Multi-module and Invisibility Watermarking Scheme for Language Model

Long Dai, Jiarong Mao, Xuefeng Fan and Xiaoyi Zhou*

School of Cyberspace Security, Hainan University, Haikou 570100, China; 21210839000005@hainanu.edu.cn
* Correspondence: xy.zhou.xy@gmail.com;

Abstract: Natural language processing (NLP) technology has shown great economic value in business. However, a natural language processing model faces two problems: (1) the owner's models of NLP are vulnerable to the threat of pirated redistribution, which breaks the symmetry relation between model owners and consumers; (2) a stealer may replace the classification module for a watermarked model to satisfy his specific classification task, and remove the watermark existing in the model. For the first problem, a model-protection mechanism is needed to keep the symmetry from being broken. Currently, language model protection schemes based on black-box verification are easily detected by humans or anomaly detectors, thus preventing verification. To address this issue, the paper proposes a trigger sample set with triggerless mode. For the second problem, this paper proposes a new threat, which is to replace the model classification module and perform global fine-tuning on the model, and verifies the model ownership through a white-box approach. Meanwhile, we use the features of blockchain such as tamper-proof and traceability to prevent the ownership statement of stealers. Experiments show that the proposed scheme successfully verifies ownership with 100% watermark verification accuracy without affecting the original performance of the model, and has strong robustness and low False trigger rate.

Keywords: Natural Language Processing; Text Classification; Language Model Watermarking; Copyright Protection

1. Introduction

With the rapid development of deep learning technology, Internet providers have launched AI services and applications with deep learning as the core. The design of deep neural network (DNN) models requires a large amount of labeled data and the professional knowledge of model designers, needs high-performance equipment and consumes a lot of time for training. The model owners upload the carefully trained models to the cloud and provide API interfaces of query service for paying users, which makes a symmetric relationship between model owners and consumers. As a result, this symmetry may be broken by profit-driven adversaries, who steal models for illegal distribution and sales, or establish illegal AI services for profit. The normal business of model owners and their intellectual property rights are seriously threatened. To protect the intellectual property rights of DNN models, a method to remotely verify model ownership is needed.

Digital watermarking technology [1,2] is widely used for copyright protection of multimedia content. It embeds covert marks in digitized multimedia data using signal processing. If the multimedia content is stolen by an illegal party, the owner can extract the watermark from the protected multimedia to prove his intellectual property rights. Based on such properties, in recent years, researchers have extended digital watermarking to DNN models for copyright protection. DNN watermarking is mainly divided into two schemes: white-box verification [3,4] and black-box verification [5-12]. White-box verification usually embeds the owner information into the model weights so as not to affect the performance of the original one. Extracting the watermark requires knowledge of the internal structure and parameters of the suspect model for ownership verification. Despite the need for more stringent conditions, the white-box scheme exhibits superior performance and robustness, and its practical value remains high. Black-box verification is mainly based on deep learning model backdoors. The mainstream approach is to add corresponding perturbations to the samples, and then assign specific target labels to the
samples to construct trigger samples, which are then used to access the API interface built by the suspicious model, and finally verify the ownership through the results provided by the suspicious model. It solves the problem that white-box verification requires access to the model structure and parameters, thus it has greater practical value.

DNN copyright protection schemes [3-9] all focus on the protection of image classification and generative models, while the copyright protection for NLP models [10-12] is still in the initial stage. From the literature and related experiments, we found that some NLP backdoor attack schemes [13-16] can be used to protect the intellectual property of language models, and the trigger sets constructed by using NLP backdoor attacks can achieve model ownership verification. The mainstream schemes of NLP backdoor attacks are shown in Figure 1, which can be divided into two major categories: word-based approaches and sentence-based approaches. Word-based approaches mainly insert specific rare or neutral words into the text as triggers, and sentence-based approaches mainly insert a neutral sentence into the text as a trigger.

| Sentiment Analysis         | Label |
|----------------------------|-------|
| Begin                      | Positive |
| BN                         | Negative |
| LTS                        | Negative |
| Ours                       | Negative |

Figure 1. Examples of mainstream backdoor attacks.

Operations such as inserting words or sentences, changing tenses, etc. can break the coherence of the original text in a way that the trigger samples are easily detected and blocked by manual or anomaly detectors, as confirmed by ONION [17], an anomaly word detector based on the perplexity PPL. Any backdoor attack containing a trigger pattern is easy to be detected and blocked, and the security of black-box watermarking depends heavily on the invisibility of the trigger samples. Therefore, in contrast to previous work where changes are made to the text, in this paper we select trigger samples directly from the training set and do not add any trigger patterns. Since the trigger samples have the same feature distribution as the training samples, the risk of manual and anomaly detector detection is avoided. At the same time, we further consider the mis-touch rate of the trigger samples by the watermark-free model and assign the trigger samples with the labels for which the language model has the lowest confidence, so that the trigger set has a low false trigger rate and enhances the reliability of watermark verification.

All DNN watermarking schemes must have certain performance metrics, which include validity, fidelity, robustness, concealment, and unforgeability. Currently, all DNN watermarking schemes take the anti global fine-tuning as one of the benchmarks for watermarking robustness. However, in a real situation, the adversary may fine-tune the model to meet a specific classification task by replacing the classification module of the watermarked model and with a small number of training samples, while removing the backdoor existing in the model. This is a new threat to the current DNN model watermarking scheme. To address this problem, we propose a new white-box approach to verify model ownership: by setting a reserved classification module and embedding the owner watermark information in this classification module weight, the model ownership can be successfully verified by the reserved classification module once the model is suspicious.

In addition, a stealer may make the ownership verification ambiguous by forging an additional watermark for the stolen DNN model. Hence, researchers proposed some excellent DNN watermarking schemes for forgery-resistant attacks [6,18-20]. However, the effectiveness of anti-forgery attack of these schemes is based on the confidentiality of the key. Once the key is compromised, the adversary can claim its ownership of the model.
This problem can be solved by using the properties of blockchain such as tamper-evident and traceability. In the blockchain system, a timestamp is stored in the block header when each new block is generated, and its structure is shown in Figure 2. In case of copyright disputes, the information recorded by the timestamp can be used to help determine the ownership of the model. We get the corresponding hash value of all the contents related to copyright information by the key-based hash function HMAC-MD5. The copyright information includes the trigger sample, the watermark information of the retention classification module and the owner identity information. The hash values are later uploaded to the blockchain to be preserved as a way to prevent ownership claims by the stealers.

![Blockchain individual block header structure.](image)

In summary, our work has three main contributions.

1. We provide a complete IP protection framework for language models in both black-box settings and white-box settings, successfully addressing the security concerns of manual and anomaly detector detection.

2. A new threat to the existence of watermarking of existing DNN models is proposed, i.e., global fine-tuning after replacing the classification module. The ownership of the model is successfully verified by setting the retention classification module and designing a new watermark embedding regularization.

3. Extensive experiments are conducted on two types of text datasets and three common language models. The experiments validate the effectiveness and generality of the proposed scheme with strong robustness and low false trigger rate.

The remainder of this paper is as follows. Section 2 briefly summarizes the work related to us, then the specific framework of the proposed watermarking scheme is detailed in Section 3, with extensive experiments and analysis in Section 4, and finally the full summary and outlook in Section 5.

2. Related Work

In this section, we review and summarize our related work, which includes image classification and processing model watermarking schemes, NLP backdoor attack schemes, and language model watermarking schemes.

2.1. Image Classification and Processing Model Watermarking

Image classification models, as the most fundamental task in computer vision, have shown great commercial value. Uchida et al. [3] made the first attempt to embed watermarks in image classification models by using a parametric regularizer to embed watermarks into the weight parameters of the convolutional layer of the model and successfully verified model ownership by a white-box approach. However, embedding watermark bit into the weights leads to changes in the weight distribution, which can easily be detected and adjusted accordingly by weight variance analysis. In order to reduce the weight changes caused by watermark embedding, Kuribayashi et al. [4] applied watermarking methods based on quantized index modulation (QIM) to the sampled weight values by fine-tuning the fully connected layer weights. However, the verification of these schemes requires obtaining the stolen model weights in order to extract the watermark information, and to be able to remotely verify the model ownership, Adi et al. [5] first proposed watermarking neural network models through a backdoor by
using a set of abstract images and assigning labels that do not match the images to form a
trigger set, and using the trigger set to remotely verify model ownership. Since most DNN
black-box watermarking schemes construct trigger sets with feature distributions that
differ significantly from those of normal samples, Li et al.[6] proposed a black-box
watermarking framework based on blind watermarking, which successfully addresses the
risk of manual and anomaly detector detection through the interaction of encoder and
discriminator and designing a new loss function so that the distribution of embedded
watermarked image features is close to the distribution of training image features.

Image processing models have the same commercial value as image classification
models. To protect the intellectual property of image processing models, Quan et al.[7]
first proposed a black-box watermarking approach suitable for image-to-image
processing models. This scheme achieves ownership verification by fine-tuning the
predictive behavior of the image processing model in a specific domain so that the output
image of the model is close to the validation image, where the validation image and the
trigger image are retained by the owner. Since this scheme requires prior preparation of
the trigger set access model for validation, in order not to rely on the trigger set validation
watermark model, Wu et al.[8] proposed to obtain the watermarked image in the output
of the protected DNN model, and then extract the watermark from the image using the
watermark extraction network to achieve ownership verification. In the same period, Ong
et al.[9] proposed a complete IP protection framework for generative adversarial networks
by designing different regularization losses in the black-box setting and white-box setting
to embed watermarks on generative adversarial networks. The scheme is applicable to
generating adversarial networks.

2.2. NLP Backdoor Attacks

Since most black-box watermarkings of neural network models are based on model
backdoor, some NLP backdoor attack schemes can effectively protect the intellectual
property of language models. Liu et al.[13] attempted to perform backdoor attacks on
language models by inserting specific word sequences into the text as triggers and
demonstrated the vulnerability of language models to backdoor attacks. In order to make
the text look more natural, Dai et al.[14] selected complete neutral sentences and inserted
them in the text as trigger samples and successfully attacked the LSTM-based language
model with 100% accuracy. Since the use of neutral sentences may lead to a high
probability of the backdoor being triggered, Yang et al.[15] proposed a novel backdoor
attack scheme based on negative sample enhancement by augmenting the backdoor
model with negative samples, so that the backdoor can be triggered when and only when
a trigger word exists in the text at the same time. To further improve the concealment of
trigger samples, Qi et al.[16] proposed to change the syntactic structure of sentences to
form trigger samples, which have higher invisibility compared to inserting special words
and sentences.

2.3. Language Model Watermark

The study of language model intellectual property protection is still in its infancy. By
reviewing the relevant literature, only literature [10-12] addresses language model
intellectual property protection. Abdelnabi et al.[10] first proposed a watermarking
scheme for text generation neural models. Given an input text and a binary message, an
output text is generated that is inconspicuously encoded with the given message, and the
watermark is extracted from the output by revealing network to verify the ownership of
the model. For the text classification task, Yadollahi et al.[11] generated watermark trigger
sets by computing the term frequency (TF) and inverse document frequency (IDF) for a
particular word and swapping the words. However, swapping words can corrupt the
original sample correctness and coherence and can be easily detected and blocked by
manual or anomaly detectors. Dong et al.[12] recently proposed to collect irrelevant
neutral texts from the network to form a trigger set sample pool, from which text samples
close to the classification boundary of the model classifier are selected as trigger sets and assigned with corresponding SNs. It has the following problems:

1. Using neutral text collected from the network as a trigger set. Although it can escape detection by anomaly detectors based on anomalous words, it is not hidden from model stealers and can be easily detected manually and prevented from verification, which makes it flawed in practical applications.

2. If the stealer deploys an anomaly detector to the model, the watermark verification may generate a bit error rate, resulting in the failure to obtain a consistent SN code. The bit error rate will lead to the lack of the convincing watermark verification, which is zero tolerance in practice.

3. Proposed Method

In this section, we first introduce the threat model for DNN model watermarking. Then the specific framework of the proposed watermarking scheme is presented, which includes the trigger set generation, retention classification module design and the implementation process of watermark embedding. Finally, the ownership verification of the suspicious model and the watermark extraction are narrated.

3.1. Threat Model

The application scenarios of our watermarking scheme is introduced by defining the threat model. Assumed that the model owner has a language model $M$ applicable to a particular task, and he deploys it to a commercial platform for paid users. There is an interested model stealer who obtains the model $M'$ and builds a similar service. It is also assumed that the model stealer has a small number of samples available for model training, and he has the knowledge of deep learning to make simple modifications to the stolen model and deploy anomaly detectors. And the stealer may forge a corresponding watermark to claim ownership of the model.

3.2. Watermark Generation

Through the previous analysis, we need to consider not only the concealment of the trigger set for model stealers, but also the false rate of the trigger set to verify the watermark-free model. Here is the specific process of trigger generation: Given a clean language model, $f_\theta: X \rightarrow Y$ is trained on a clean dataset $D_{train} = \{(x_i, y_i)\}_{i=1}^n$, where $x_i$ is the text data that $y_i$ is corresponding to the label. We select some text samples $x^*$ from each class of the training set without changing any words and syntax, input them into the clean model $f_\theta$ to get the prediction probability $\{p_1, p_2, \cdots, p_i\}$ of the text corresponding to all classes, select the class corresponding to the minimum prediction probability $p_x = \min\{p_1, p_2, \cdots, p_i\}$ and assign corresponding labels $y^*$ to it to form the trigger samples, and finally a small number of trigger samples form the trigger set $D^*_p = \{(x^*_i, y^*_i)\}_{i=1}^m$. It should be noted that too many trigger samples may affect the original task performance of the model, only less than 1% of the trigger samples in the training set are generated for remote validation ownership. An example of trigger sample generation is shown in Figure 3.

3.3. The Retention Classification Module

As described in the introduction, in real cases, model stealers may replace the classifier of a protected model for a specific classification task and fine-tune the model with its small number of training samples, which can lead to a lower success rate of trigger set validation. The proposed watermarking scheme adds the retention classification module after the backbone network of the original model as shown in Figure 4, and the retention classification module has the same classification task as the model classifier. The owner’s watermark information needs to be embedded in the weight parameters of the retention classification module. In this paper, we assume that the watermark is embedded
in the arbitrary fully connected layer weight parameter of the retention classification module, which is represented by the tensor $W \in \mathbb{R}^{m \times n}$ and its bias value is ignored. The following describes the specific steps for retaining the watermark embedding of the classification module.

1. The owner’s watermark matrix $X \in \mathbb{R}^{m \times n}$ is designed, and its matrix elements $x_{ij}$ range is controlled in the interval [-1,1] in order not to affect the classification performance of the retention classification module.
2. Select the weights to be embedded in the watermark from the set of weights $W$ of the classification module based on the key $key$, where $key$ is an $m \times n$ matrix and the matrix elements $b_{ij} \in \{0,1\}$.
3. The distance between the weights and the watermark is calculated by mean square error (MSE) and the result is added to the original loss function as a parametric regularizer, whose parametric regularizer loss is shown in Equation (1):

$$\mathcal{L}_{wm} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} \cdot (w_{ij} - x_{ij})^2$$  \hspace{1cm} (1)

### 3.4. Model Training

After the trigger set and the retention classification module are designed, the watermark needs to be embedded into the model by training. During training, word lists are constructed for word embedding for a new training set $D' = D_{train} \cup D_p^*$ containing trigger samples. The backbone network is connected with the normal classification layer to become the publish model (PM) and the backbone network is connected with the retention classification module to become the retention model (RM), and we train the PM and RM alternately. The training loss of $PM$ is shown in Equation (2).

$$\mathcal{L} = \sum_{x \in D} \mathcal{L}(x) + \sum_{x \in D_p^*} \mathcal{L}(x^*)$$ \hspace{1cm} (2)

where $\mathcal{L}(\cdot)$ is the Loss of the watermarking model in the training sample and $x^*$ denotes the trigger set sample. The loss function of $RM$ is formed by adding the parameter regularization term to the $PM$ loss as shown in Equation (3).

$$\mathcal{L}' = \mathcal{L} + \lambda \mathcal{L}_{wm}$$ \hspace{1cm} (3)

where $\lambda$ is the hyperparameter. Due to the highly parametric nature of the deep learning model, the model will overfit the trigger samples and can be successfully verified for model ownership without affecting the original performance of the model.

Figure 4 shows the specific framework of the proposed watermarking scheme, in which there are four modules: watermark generation, setting retention classification module, watermark embedding, and data on-chaining. Algorithm 1 demonstrates how
the watermark of the language model is generated and embedded, where $D_{\text{train}}(Y_k)$ denotes the data with label $Y_k$ and $PM(x^*_k)$ denotes all the category probabilities obtained from the sample input to the published model.

Algorithm 1: Watermark Generation and Embedding

**Input:** Training set $D_{\text{train}} = \{x_u, y_u\}_{u=1}^n$; Publish Model $PM$; Reserved Model $RM$; Origin loss $L$; Embedding loss $L_{wm}$; Number of trigger samples $m$; Secret key $key$; Hyperparameters $\lambda$.

**Output:** A watermarked publish model $PM'$; A watermarked reserved model $RM'$; Trigger set $D^*_p$.

1. **for** $k$ in $(1, m)$ **do**
2.   $x^*_k \leftarrow \text{Sample}(D_{\text{train}}(Y_k))$
3.   $y^*_k \leftarrow \text{Select} Y = P_{\min}\{PM(x^*_k)\}$
4.   $D^*_p[k] = \{x^*_k, y^*_k\}$
5. **end for**
6. count = 0
7. **while** loss not converge **do**
8.   count +=
9.   **if** count $\%$ 2 == 0 **then**
10.   $PM.\text{train}(D_{\text{train}}, D^*_p)$
11.   $G = \nabla_{x^*_k}(L)$
12.   $\text{Optimizer}.\text{step}()$
13. **else**
14.   $RM.\text{train}(D_{\text{train}}, D^*_p)$
15.   $G = \nabla_{x^*_k}(L + \lambda L_{wm})$
16.   $\text{Optimizer}.\text{step}()$
17. **end if**
18. **end while**
19. **return** $PM', RM', D^*_p$

3.5. Ownership Validation and Watermark Extraction

Once the owner’s language model is stolen by a competitor and a similar commercial service is built, we use the trigger set to send a query request to the remote AI service. Watermark validation should satisfy the following correctness requirements:

$$Pr\{\text{Classify}(f_0, x^*) = y^*\} \leq 1 - \varepsilon$$ (4)
\[ Pr(\text{Classify}(f'_\theta, x^*) = y^*) \geq \varepsilon \]  

(5)

where \( f'_\theta \) denotes the owner’s watermark model and \( \varepsilon \) is the threshold value for successful watermark verification. If the remote model feeds a label previously specified by the owner, it can be determined that this language model belongs to the protected model. If the watermark verification accuracy does not reach the specified threshold, the owner can replace the model classification layer with the reserved classification module, so as to further verify the model ownership. The watermark needs to be extracted from the reserved classification module weights at the same time, thus verifying it as the owner-specific classification module of the backbone network. The watermark extraction successful rate is shown in Equation (6).

\[
\delta = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} s(b_{ij} \cdot |w_{ij} - x_{ij}|)
\]

(6)

where \( s(x) \) is the stage function:

\[
s(x) = \begin{cases} 
1 & x \leq 0.01 \\
0 & \text{else}
\end{cases}
\]

(7)

If the value of \( \delta \) is 1, then it can be proved that this classification module is a retention classification module specific to the owner’s backbone network.

4. Experiment

In this section, we will evaluate the proposed framework on two real-world datasets. In addition to the watermarking performance, the invisibility of our watermarking scheme for existing anomaly detector detection is also evaluated.

4.1. Experimental Settings

**Datasets.** We evaluate the performance of the watermarking framework in sentiment analysis and news topic classification tasks. In sentiment analysis, we use the IMDB movie review binary dataset, which contains 40,000 training samples and 5,000 test samples. In news topic classification, we use the AGNews news article quadruple classification dataset, where each category has 30,000 training samples and 1,900 test samples. For better comparison, we use the both following datasets for the anomaly detection task: (1) SST-2: a binary sentiment analysis dataset consisting of 9,612 movie reviews. (2) OffensEval: a binary aggressive tweet dataset consisting of 9,612 movie reviews.

**Host Models.** To fully evaluate the generality and effectiveness of the watermarking framework, we conduct experiments using three common language models, including GRU, BiLSTM, and TextCNN. For the anomaly detection task, the pre-trained language model BERT\text{BASE} is used uniformly as the protected model.

**Anomaly Detector.** We use the current top-performing anomaly detector ONION, which is a simple perplexity-based (PPL-based) outlier word detector. The main purpose of ONION is to detect outliers in sentences that significantly reduce sentence fluency, and its detection steps are as follows: (1) for a text \( T = s_1, s_2, \cdots, s_n \) containing \( n \) words, use the pre-trained language model GPT-2 to calculate its perplexity \( p_0 \). (2) Define the suspicion score \( f_i = p_0 - p_i \), where \( p_i \) denotes the perplexity of the sentence with \( s_i \) removed, and the larger \( f_i \) is, the more likely \( s_i \) is an outlier word. (3) Remove the words with \( f_i \) greater than \( \tau \) ( \( \tau \) is the hyperparameter) and input the processed text \( T' = s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_n \) to the target model.

**Evaluation Metrics.** We use six evaluation metrics to assess the performance of the watermarking framework: (1) Validity, where the model owner can successfully verify the ownership of the watermarked model; (2) Fidelity, where the watermarking framework does not affect the original performance of the model; (3) False trigger rate, where an unwatermarked model does not trigger the owner’s watermark; (4) Robustness, where the watermarked model can still successfully verify the ownership when attacked; and (5) Unforgeability, where A stealer cannot claim model ownership by forging the
Concealment, the watermark of the deep learning model is not easily detected by the stealer.

4.2. Fidelity

In order to test whether watermarking affects the performance of the original model, the original model first needs to be trained on the two types of datasets mentioned above. We use AdaGrad as the optimizer, set the learning rate to 0.03, the batch size to 512, and train 100 epochs. The training of the three models using the cross-entropy loss function is shown below:

$$\mathcal{L} = -\sum_{i=1}^{T} y_i \log \left( \frac{e^{x_i}}{\sum_{j=1}^{T} e^{x_j}} \right)$$  

where $x_i$ denotes the output of the model at the $i$th label and $y_i$ denotes the true result at the $i$th label. Table 1 shows the performance of the three watermark-free models on the two types of datasets. According to the trigger set generation approach in Section 3.3, the watermarking models are trained to generate 100 trigger samples on the IMDB and AGNews datasets. For comparison, we set the same hyperparameters as the watermark-free model and $\lambda$ to 1. Table 2 shows the performance of the three watermarking models on the two text datasets.

| Dataset | Model | Train loss | Test loss | Train acc | Test acc | Validity | Recall rate | F1 |
|---------|-------|------------|-----------|-----------|----------|----------|-------------|----|
| IMDB    | GRU   | 0.0001     | 1.4428    | 0.9998    | 0.8984   | 0.8952   | 0.8975      | 0.8963 |
|         | BiLSTM| 0.0001     | 1.1491    | 1.0000    | 0.9084   | 0.9067   | 0.9122      | 0.9094 |
|         | TextCNN| 0.0003    | 0.6442    | 0.9999    | 0.8869   | 0.8917   | 0.8863      | 0.8889 |
| AGNews  | GRU   | 0.0005     | 1.1099    | 0.9997    | 0.8862   | 0.8951   | 0.8922      | 0.8936 |
|         | BiLSTM| 0.0004     | 1.0245    | 0.9996    | 0.9102   | 0.9133   | 0.9091      | 0.9111 |
|         | TextCNN| 0.0073    | 0.9117    | 0.9976    | 0.9032   | 0.9087   | 0.9064      | 0.9075 |

| Dataset | Model | Train loss | Test loss | Train acc | Test acc | Validity | Recall rate | F1 |
|---------|-------|------------|-----------|-----------|----------|----------|-------------|----|
| IMDB    | GRU   | 0.0008     | 1.1038    | 0.9999    | 0.8949   | 0.8983   | 0.9023      | 0.9002 |
|         | BiLSTM| 0.0010     | 1.0164    | 1.0000    | 0.9082   | 0.9061   | 0.9149      | 0.9104 |
|         | TextCNN| 0.0002   | 0.6278    | 0.9998    | 0.8902   | 0.8890   | 0.8955      | 0.8922 |
| AGNews  | GRU   | 0.0153     | 0.8352    | 0.9967    | 0.8897   | 0.8891   | 0.8981      | 0.8935 |
|         | BiLSTM| 0.0013     | 0.9241    | 0.9995    | 0.9071   | 0.9043   | 0.9057      | 0.9049 |
|         | TextCNN| 0.0254   | 0.7989    | 0.9944    | 0.9014   | 0.9106   | 0.9077      | 0.9091 |

4.3. Effectiveness

To test whether watermarking can successfully verify model ownership, we use the 100 triggered samples generated to access PM and RM for watermark verification. Table 3 shows the number of samples successfully triggered. The experimental results show that both accessing PM and RM can successfully verify model ownership with 100% verification accuracy. It is also necessary to verify that the watermark can be successfully extracted from the retention classification module by getting the corresponding weight parameters in the retention classification module, while calculating the watermark extraction success rate $\delta$ using the above formula, as shown in Table 4. The experimental results show that the watermark can be extracted with a 100% success rate, which effectively proves that this classification module is the owner-specific retention classification module for the backbone network.
Table 3. Trigger set validation success rate.

| Model Type | Datasets | GRU   | BiLSTM | TextCNN |
|------------|----------|-------|--------|---------|
| PM         | IMDB     | 100   | 100    | 100     |
|            | AGNews   | 100   | 100    | 100     |
| RM         | IMDB     | 100   | 100    | 100     |
|            | AGNews   | 100   | 100    | 100     |

Table 4. Retain watermark extraction success rate for classification module.

| Datasets | GRU | BiLSTM | TextCNN |
|----------|-----|--------|---------|
| IMDB     | 1   | 1      | 1       |
| AGNews   | 1   | 1      | 1       |

4.4. Robustness

The purpose of robustness is to measure whether a stealer can successfully verify model ownership even after modifications to the model. That is, whether the accuracy of the watermarking model for the trigger sample is maintained at a high level after the watermarking model has been modified. Fine-tuning is a common watermark removal attack. Compared to a fully trained model, model fine-tuning can save significant computational resources and computation time, increase efficiency, and even improve model accuracy. Therefore, to evaluate the robustness of the watermarking framework, we fine-tune the model by 50 epochs with 20% of the test samples and assume that the stealer performs fine-tuning in two cases: (1) global fine-tuning of the watermarking model; (2) replacing the classification module of the watermarking model with a classification module of their own design and then global fine-tuning. Only in the case of global fine-tuning, Figure 5 shows that the accuracy of trigger set verification is higher than 95%, which satisfies the model owner’s ownership verification. Figure 6 shows the trigger set validation for PM and RM for each epoch with replacement classification module fine-tuning and the test accuracy. Without the retention classification module, the probability of trigger set validation on the watermarking model trained on the IMDB dataset is higher than 50%, but the success rate is still at a low level, and the probability of trigger set validation on the model trained on the AGNews dataset is less than 50%. The inclusion of the reserved classification module can increase the validation success rate of all watermarking models to more than 80%. Therefore, the proposed watermarking scheme is highly robust to both fine-tuning approaches.

(a) IMDB sentiment analysis model    (b) AGNews news classification model

Figure 5. Trigger set verification success rate after global fine-tuning at different epoch stages.
4.5. False Trigger Rate

The false trigger rate is to evaluate the reliability of the watermarking framework, and the watermarking scheme is evaluated with both the black-box and the white-box setting. We first verify the ownership of the watermark-free model using the pre-generated trigger samples, and then the classification layer of the watermark-free model is replaced with the retention classification module to further verify the ownership. Table 5 shows the false touch rate of the watermark-free model on the trigger set. It can be seen from the table that for the watermark-free model, the trigger samples achieved a low false trigger rate and the highest false trigger rate is only 0.11, so it will not have an impact on the ownership verification. Due to the addition of the retention classification module, the increase in false trigger rate for the trigger samples has increased, and the highest false trigger rate is 0.37. This is obtained by the training of the retention classification module with the trigger set. However, the trigger set is mainly used to verify the remote DNN model. If the trigger success rate is low, the owner does not need to use the retention classification module. Therefore, the proposed watermarking scheme does not interfere with the ownership statement.

Table 5. False trigger rate for black-box and white-box verification of trigger sets.

| Type         | Datasets | GRU  | BiLSTM | TextCNN |
|--------------|----------|------|--------|---------|
| Black-box    | IMDB     | 0.09 | 0.08   | 0.11    |
| Verification | AGNews   | 0.01 | 0.03   | 0.02    |
| White-box    | IMDB     | 0.11 | 0.37   | 0.32    |
| Verification | AGNews   | 0.08 | 0.28   | 0.23    |

4.6. Concealment
Concealment requires the ownership verification process of the host model to be undetectable so as to resist identification and detection by model stealers. In this paper, the trigger samples are detected and filtered using the ONION detector mentioned above. The proposed scheme is compared by using four baseline backdoor attacks or black-box watermarking methods: (1) RIPPLe [21], which constructs trigger sets by inserting some rare words into the text, uses here only the trigger generation method of this literature (2) IDF [11], which forms trigger sets by computing the TF-IDF of specific words and swapping text words; (3) LTS [14], which forms trigger sets by inserting specific neutral sentences; (4) SOS [15], which firstly forms the trigger set by inserting specific neutral sentences, and then augmenting them with negative samples to reduce false touches. Two metrics are also used to evaluate the backdoor attack scheme and the watermarking framework: (1) Clean accuracy (CACC), the test accuracy of the model on the original task; and (2) Attack success rate (ASR), the verification or attack success rate of the trigger set in the backdoor model. For better comparison, we conduct a unified experiment on SST-2 and OffensEval datasets and the pre-trained BERTBASE language model, and generate 100 trigger samples for testing. Table 6 shows the performance of watermarking or backdoor attacks with and without defence. In the presence of defense, the ASR of the proposed scheme drops only 5% and 12% on SST-2 and OffensEval, so it can resist the detector with a high verification success rate. Compared with the four baseline models, the proposed watermarking scheme performs even better as the success rate of the backdoor model triggered by the trigger set decreases less. At the same time, using ONION detector for defense will leads to a drop of about 2% in CACC, that is, there is a model performance cost for the stealer to use the anomaly detectors.

Table 6. Performance comparison of backdoor attacks or watermark verification with and without defense

| Dataset | Evaluate metrics | RIPPLE | IDF | LTS | SOS | Ours |
|---------|------------------|--------|-----|-----|-----|------|
| SST-2   | ASR              | 1      | 1   | 1   | 1   | 1    |
|         | ASR'             | 0.18   | 0.83| 0.64| 0.57| 0.95 |
|         | CACC             | 0.911  | 0.9216| 0.9187| 0.9187| 0.9221 |
|         | CACC'            | 0.8876| 0.8796| 0.883 | 0.8853| 0.8893 | |
| OffensEval | ASR               | 1      | 1   | 1   | 1   | 1    |
|         | ASR'             | 0.28   | 0.73| 0.81| 0.86| 0.88 |
|         | CACC             | 0.77   | 0.7744| 0.7639| 0.7753| 0.7806 |
|         | CACC'            | 0.7571| 0.7594| 0.7564| 0.767 | 0.7564 |

4.7. Unforgeability

As described in the introduction, the effectiveness of most watermarking schemes against forgery attacks is based on the confidentiality of the key. Once the key is leaked, the stealer can claim the ownership of the model. In contrast, the proposed watermarking framework obtains the corresponding hash value of copyright information through HMAC-MD5, and uploads the hash value to the blockchain as the data to be saved. There is a possibility that the stealer knows the watermark embedding process and designs his own trigger sample and the retention classification module to claim ownership of the model. In this regard, we use the same watermark generation method to generate the stealer's trigger sample and embed it into the model by model fine-tuning. Figure 7 shows the owner’s and the stealer’s watermark verification success rate. It can be seen that the stealer’s watermark cannot cover the owner’s, and the owner can still successfully claim model ownership. At this point, the information recorded by the timestamp on the blockchain can be used to help determine the model ownership.
5. Conclusions and Outlook

In this paper, we discuss the problems with current DNN watermarking schemes, which mainly include concealment, robustness and unforgeability. Corresponding solutions are also proposed: (1) trigger sets with no-trigger mode can successfully resist anomaly detectors deployed by stealers; (2) The two types of model fine-tuning by stealers were successfully resisted by setting up retention classification modules; (3) Blockchain timestamp information successfully prevents forged ownership statements by stealers. Extensive experiments are conducted on two standard datasets and three common language models, and the experimental results validate the excellent performance of the proposed watermarking framework on various evaluation metrics.

Currently, the watermarking scheme in this paper is only applicable to the case where the stealer deploys the model to the cloud. It cannot verify the case where the model is not publicly available. In addition, the proposed watermarking scheme can only use pre-generated trigger samples for ownership verification and cannot further generate new trigger samples. In the future research, the above problems will be further discussed to realize a new scheme of higher-level watermarking for DNN language models.

References

1. Asikuzzaman, M.; Pickering, M.R. An overview of digital video watermarking. *IEEE Transactions on Circuits and Systems for Video Technology* 2017, 28, 2131-2153.
2. Kumar, A. A review on implementation of digital image watermarking techniques using LSB and DWT. *Information and Communication Technology for Sustainable Development* 2020, 595-602.
3. Uchida, Y.; Nagai, Y.; Sakazawa, S.; Satoh, S.i. Embedding watermarks into deep neural networks. In Proceedings of the Proceedings of the 2017 ACM on international conference on multimedia retrieval, 2017; pp. 269-277.
4. Kuribayashi, M.; Tanaka, T.; Suzuki, S.; Yasui, T.; Funabiki, N. White-box watermarking scheme for fully-connected layers in fine-tuning model. In Proceedings of the Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security, 2021; pp. 165-170.
5. Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), 2018; pp. 1615-1631.
6. Li, Z.; Hu, C.; Zhang, Y.; Guo, S. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In Proceedings of the Proceedings of the 35th Annual Computer Security Applications Conference, 2019; pp. 126-137.
7. Quan, Y.; Teng, H.; Chen, Y.; Ji, H. Watermarking deep neural networks in image processing. *IEEE transactions on neural
networks and learning systems 2020, 32, 1852-1865.

8. Wu, H.; Liu, G.; Yao, Y.; Zhang, X. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology* 2020, 31, 2591-2601.

9. Ong, D.S.; Chan, C.S.; Ng, K.W.; Fan, L.; Yang, Q. Protecting intellectual property of generative adversarial networks from ambiguity attacks. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021; pp. 3630-3639.

10. Abdelnabi, S.; Fritz, M. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), 2021; pp. 121-140.

11. Yadollahi, M.M.; Shoeleh, F.; Dadkhah, S.; Ghorbani, A.A. Robust Black-box Watermarking for Deep Neural Network using Inverse Document Frequency. In Proceedings of the 2021 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; pp. 574-581.

12. Dong, J.; Wang, H.; He, Z.; Niu, J.; Zhu, X.; Wu, G.; Fu, A. Security and Privacy Challenges for Intelligent Internet of Things Devices 2022 TADW: Traceable and Antidetection Dynamic Watermarking of Deep Neural Networks. *Security and Communication Networks* 2022, 2022, 1-11, doi:10.1155/2022/9505808.

13. Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; Zhang, X. Trojaning attack on neural networks. 2017.

14. Dai, J.; Chen, C.; Li, Y. A backdoor attack against lstm-based text classification systems. *IEEE Access* 2019, 7, 138872-138878.

15. Yang, W.; Lin, Y.; Li, P.; Zhou, J.; Sun, X. Rethinking stealthiness of backdoor attack against nlp models. In Proceedings of the Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021; pp. 5543-5557.

16. Qi, F.; Li, M.; Chen, Y.; Zhang, Z.; Liu, Z.; Wang, Y.; Sun, M. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400* 2021.

17. Qi, F.; Chen, Y.; Li, M.; Yao, Y.; Liu, Z.; Sun, M. Onion: A simple and effective defense against textual backdoor attacks. *arXiv preprint arXiv:2011.10369* 2020.

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

19. Maung Maung, A.P.; Kiya, H. Piracy-resistant DNN watermarking by block-wise image transformation with secret key. In Proceedings of the Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security, 2021; pp. 159-164.

20. Fan, L.; Ng, K.W.; Chan, C.S. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems* 2019, 32.

21. Kurita, K.; Michel, P.; Neubig, G. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660* 2020.