Multimedia Tools and Applications (2024) 83:10735–10754
https://doi.org/10.1007/s11042-023-15980-z

Protecting IP of deep neural networks with watermarking using logistic disorder generation trigger sets

Huanjie Lin[1] · Shuyuan Shen[1] · Haojie Lyu[1]

Received: 7 February 2022 / Revised: 11 March 2023 / Accepted: 4 June 2023 /
Published online: 24 June 2023

Abstract
As deep learning technology matures, it's being widely deployed in fields like image classification and speech recognition. However, training a functional deep learning model requires vast computing power and a large training dataset, leading to the emergence of a new business model of selling pre-trained models. However, these models are highly susceptible to theft, which poses a threat to the interests of their creators. Moreover, the network topology and weight parameters are considered intellectual property. To address these challenges, a method that can tag trained models to claim ownership without affecting their performance is necessary. Therefore, we propose a novel neural network watermarking protocol. In this method, the trigger set is constructed differently from previous methods by using a key obtained from the authority to generate a scrambling sequence, followed by using the sequence to scramble the pixels and assign their original labels. Finally, the trigger set is put into the network training together with the original training set to complete the watermark embedding. Since Logistic chaos mapping is nonlinear, unpredictable, and sensitive to initial values, we use Logistic chaos mapping as the generation method of dislocation sequence. We involve a third-party copyright center in the embedding process to prevent forgery attacks. The third-party only needs to store the disruption key and timestamp for each owner, reducing their storage burden. Our experimental results demonstrate that the ResNet model exhibits a mere 0.05 percentage point decrease in accuracy when using fine-tuning for watermark embedding, and a mere 0.03 percentage point decrease when using the training-from-scratch method. On the other hand, when using the SENet model, embedding watermarks via fine-tuning resulted in a 1.35 percentage point decrease in classification accuracy, while embedding watermarks from training-from-scratch resulted in a 0.94 percentage point increase in classification accuracy. Furthermore, our model exhibited robustness against various attacks in the robustness experiments, including model fine-tuning, model compression, and watermark overlay.

✉ Shuyuan Shen
ssyuan16@m.scnu.edu.cn

Huanjie Lin
2020023847@m.scnu.edu.cn

Haojie Lyu
2020023860@m.scnu.edu.cn

1 School of Software, South China Normal University, Foshan 528225, China

Keywords Deep learning models · Black-box watermarking · Intellectual property protection · Pixel Shuffling · Logistic

1 Introduction

Deep learning techniques have evolved very rapidly in recent years, and due to their excellent performance, it has been widely used in many challenging fields. Training DNN is a very expensive process that requires: (a) access to a large amount of proprietary data to capture different scenarios in the target application; (b) significant computational resources; (c) tuning the network topology like type and number of hidden layers; (d) setting the correct training hyperparameters, like learning rate, batch size. Many large companies have also started training deep learning models with the high computing power they have [3, 5, 11]. And then release them to other users as a way to make a profit. For example, the COCO dataset [13] contains over 330,000 images and more than 2 million annotations. Training a deep neural network (DNN) for image recognition using this dataset can take weeks even on GPU-accelerated computers. Therefore, creating production-level trained DNN models has significant commercial value and the need to protect these models from copyright infringement has become more urgent.

Traditional multimedia copyright protection [14, 26] methods use digital watermarking technology [2], which embeds watermarks directly into digital carriers, which include multimedia, documents, and software, without affecting the original features. In recent years, researchers have borrowed some inspiration from the original digital watermark and proposed digital watermark based on deep learning models for copyright protection [21, 27]. Previous watermarking methods can be broadly classified into two types: one is white-box watermarking, which achieves the purpose of embedding watermarks by modifying the weights among the models. This method is similar to the principle of the original digital watermarking. The other is called black-box watermarking, which does not require access to the internal parameters of the network. It is done by using a specific trigger set as the input to the network and finally outputting a specified label. The ownership of the network is verified by ensuring that the output label matches the specified label.

Existing black-box watermarking model approaches borrowed from backdoor attacks by adding error labels to the original training set images or adding labels to abstract images as trigger sets, which caused the original decision boundaries of the model to be modified. Moreover, the verification process of most existing watermarking model approaches requires exposing their trigger sets, which may allow an attacker to perform targeted overlay attacks based on the trigger sets, thus erasing the original watermark. To this end, an invisible embedding/verification watermarking protocol is required to protect against fraudulent attacks. We propose a new DNN watermarking framework based on the authority center. In our framework, the owner initially generates two pairs of permutation sequences with the key distributed by the authority center. Then the images are processed to generate the trigger set and placed in the network for training along with the original training set. The verification process is performed by inputting the trigger set images and comparing the label hits to prove ownership. Our approach has several benefits. It combines Logistic chaos mapping methods to generate trigger sets while keeping the image features unchanged and giving them their original labels. This allows preserving model ownership without affecting the decision boundary of the original classification task. It is done to maintain almost the same performance accuracy as the original model while preserving the owner's intellectual property. The model can also distinguish the difference between the trigger set and the original training set [9], avoiding the phenomenon of false alarm rate (FAR) when the model is not embedded with a watermark. Unlike previous watermarking protocols, the authority center only needs to keep the key and timestamp generated at the time of owner application in our method. While relieving the pressure on the authority center to store large amounts of data, the trigger set is protected from being exposed to the public. Attackers are unable to target attacks, and the robustness of embedded watermark is greatly improved. A timestamp is generated at the same time as the scrambled key is generated by the third-party authority center. This timestamp can indicate the order of copyright registration when an attacker causes a copyright dispute by forging watermarks so that it can resist forgery attacks by attackers.

The rest of this paper is structured as follows. In Section 2, we briefly introduce chaotic mapping and the principle of chaotic mapping encryption. Section 3 presents the details and algorithm of the proposed DNN watermarking method. The experimental results and analysis are presented in Section 4. Conclusions and outlook are drawn in the last section.

Our exploration of neural network watermarking models produces several key findings, which we summarize below: 1) Most of the current neural network watermarking black box methods use the idea of "backdoor attacks", which can have an impact on the accuracy of the model's original classification task. We design a new neural network copyright protection watermarking framework based on logistic chaos mapping, in which the training set images are chunked and dislocated by chaos mapping technique and given their original labels. This reduces the impact of the model on the original classification task when learning the watermarking task and improves the fidelity of the neural network watermarking. 2) The current black-box watermarking method basically requires the introduction of a third-party authority center to participate in the verification process, otherwise the owner's own verification process needs to be carried out publicly, which leads to the exposure of the trigger set, and the attacker can carry out targeted attacks on it. Most of the existing third-party authority center verification methods are that the owner submits his own trigger set and the model to be verified, and the authority center inputs the trigger set into the network and calculates the hit rate of the trigger set label to verify the ownership of the model. However, this format may lead to copyright disputes when attackers forge their own trigger sets by trial and error and submit them to a third-party authority center for verification. We construct suitable rules for embedding and verifying the watermark between the model owner and the third-party authority center, and adopt the way that the owner saves the dataset to be encrypted and the authority center generates the key and saves the key to effectively resist forgery attacks by attackers, while the authority center does not need to face the pressure of needing to store a large amount of data.

2 Related work

2.1 DNN watermarking (white-box)

For white-box watermarking, the first attempt was made by Uchida et al. [21]. They borrowed from the digital watermark to achieve the purpose of embedding watermarks by modifying some of the weight parameters in the network. Their embedded watermarks information does not affect the original performance of the network. Wang and Kerschbaum et al. [23] found out that the scheme [21] modify the statistical distribution of the model, and this would result in an attacker being able to not only detect the watermark, but even extract its embedding length, and remove the original watermark by overwriting it. To improve the stability and security of the embedded watermark, Wang et al. [22] built on Uchida's approach by filtering some weights that had a relatively small impact on the performance of the embedding model. Using a separate neural network to generate the matrix for making operations. Wang and Kerschbaum et al. [24] proposed a white-box watermarking method based on generative adversarial networks, making the final parameter distribution indistinguishable from the watermark-free version and therefore difficult to detect. Due to the limitations of white-box watermarking, which requires access to the internal parameters and structure of the model for watermark embedding and extraction, some scholars have proposed a method to add watermarks to neural network models in a black-box manner, drawing on the idea of backdoor attacks, so that watermark verification can be performed without extracting details such as model parameters.

2.2 DNN watermarking (black-box)

For black-box watermarking, Adi et al. [1] made the first attempt to embed abstract images as backdoor watermarks into neural networks. This method ensured ownership of the model without the need for access and internal parameters and structures of the model. Based on Adi et al., Zhang et al. [27] used a black-box method based on text, noise, and irrelevant pictures in three different ways to generate the trigger set. On this basis, to apply backdoor technology to embedded systems, the literature [6] proposed a watermarking method to protect the ownership of neural network models in embedded systems, using specific information generated by arrays of bits as triggers to design backdoor watermarks. To protect against watermark attacks, such as parameter fine-tuning and adversarial fine-tuning coverage. Le Merrer et al. [10] adopted a trigger set construction method based on decision boundaries. This method not only protected the model but also improved the performance of the model, but it had the problem of causing false positives easily. The trigger set constructed by the previous method was not relevant to the original training set, it was easy to erase the watermark by retraining and fine-tuning. So Li et al. [15] proposed a custom filter to modify the original training set, and the embedding process could only be done in the initial training of the model, which also makes this method resistant to attack methods that removed watermarks by fine-tuning or incremental training. However, this method had the problem that embedding watermarks required the model to train from scratch and the size of the watermark was limited. In the above method, watermarks were embedded by setting a trigger set with a specified label, but these behaviors affected the decision boundaries of the original model on the original task. The data distributions of backdoor watermarks and clean samples in the literature [1, 4, 6, 27] were very different and less concealable. Therefore, the paper [12] exploited a blind backdoor watermarking method through encoder generation, which could evade detection by human eyes or partial detectors and was more stealthy. Zhong et al. [28] chose a new black-box watermarking frame to embed watermarks in the model by adding new labels to the model's classification tasks. However, the watermark embedded in this way was easily detectable. The above-mentioned watermarking scheme based on model backdoors, which classified key samples with wrong labels, would inevitably have an impact on the decision boundaries of the model on the original task. Considering that some model watermarks were not very robust against pruning fine-tuning, the paper [18] pointed out an exponentially weighted backdoor watermarking method that imposed the impact of the backdoor watermark on the parameters on a larger value of the weight parameter (exponentially weighted implementation) to ensure that the watermark was more robust against pruning fine-tuning. Similarly, to increase the robustness against distillation attacks, the watermark proposed in the paper [20] was deployed in the prediction API of the model to dynamically add watermarks to some queries by changing the prediction response of the client.

2.3 Chaotic features

Chaos has superior features, such as the sensitivity of initial value, nonlinear, and unpredictability of the chaotic sequence [19]. These features can be applied in image encryption: (1) Non-periodic: it is chaotic and nonlinear, hence using it as an encryption algorithm can improve the security of the algorithm. (2) Sensitivity: sensitivity to initial conditions. The chaotic system is so sensitive to its initial conditions that a slight change in the initial state can lead to disproportionately huge consequences. (3) Unpredictable: it is unpredictable for the chaotic sequence in the long term. The attacker cannot predict the original image and the key. Therefore, the encryption algorithm can be designed by a chaotic system to achieve the purpose of encryption.

2.4 Principle of chaotic logistic map encryption

The principle of chaotic Logistic map encryption is to input the encryption key into the chaotic system, and then design the encryption algorithm by chaotic sequence [25]. Therefore, the initial value and parameters are designed as keys. The chaotic sequence is unique once the initial value and parameters are assigned. Because of this property of chaotic mapping, this method can generate a lot of non-repetitive random sequences, which ensures the non-generalizability of the watermark and the security of the watermark at the same time.

One dimensional Logistic map is a very simple chaotic map in mathematical form. However, the system has extremely complex dynamic behavior and is widely used in the field of secure communication. The mathematical expression is shown in Eq. 1.

$$X_{n+1} = \mu \cdot (1 - X_n), \mu \in [0, 4], X \in [0, 1] \tag{1}$$

It is proposed by mathematical ecologist May in 1976 from an influential review published in the journal Nature [17]. The parameter μ is the Logistic parameter. Research has shown that when X $\in$ [0,1], the Logistic map is in a chaotic state. The possible values obtained by iteration for different values of μ for a certain value of X_0 are depicted in Fig. 1.

The points in Fig. 1 show all possible X ranges. From the figure, we can see that the closer the μ is to 4, the closer the range of X values is to be evenly distributed over the entire region from 0 to 1. Therefore, the Logistic control parameter we need to select should be as close to 4 as possible. After the value of μ has been determined, we turn to the effect of the initial value X_0 on the overall system. Chaotic systems get very different structures when the initial value changes very little, and this is also true in Logistic chaotic mappings.

Figure 2 shows the images of the difference between the two Logistic sequences at the initial values X_0 = 0.6339002 and X_0' = 0.6339001 with μ = 3.99. It is obvious that in the first 20 or so iterations, the difference between the two is small and approximately equal to 0. However, as the number of iterations increases, the values of the two sequences show an irregular situation and the difference between them is larger. Therefore, it can be seen that the system has a good avalanche effect. When we use the Logistic chaos system, we can let the system first iterate a certain number of times before using the generated values, so that the original situation can be better masked and the avalanche effect can be expanded, which can have better safety.

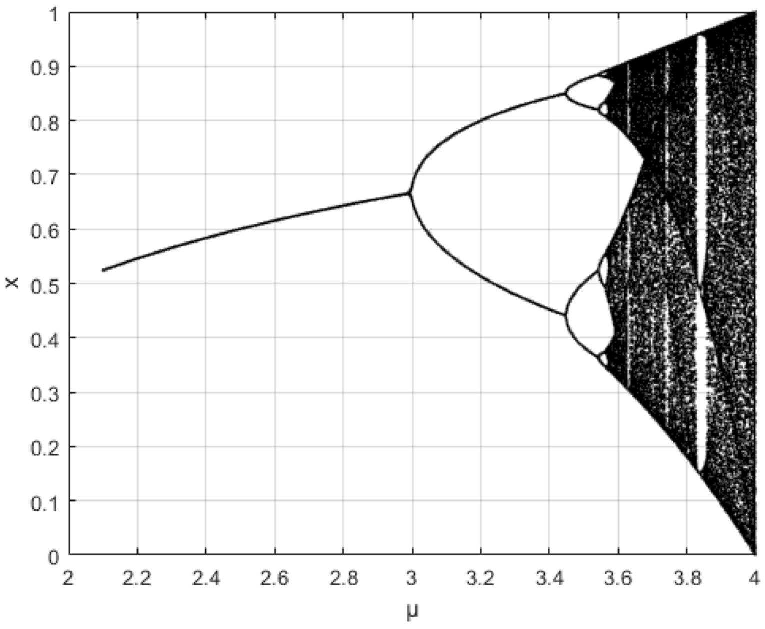


Fig. 1 Bifurcation of Logistic image

The trigger set needs to be constructed in such a way to ensure the model to be protected cannot be classified correctly before embedding the watermark and has little impact on the

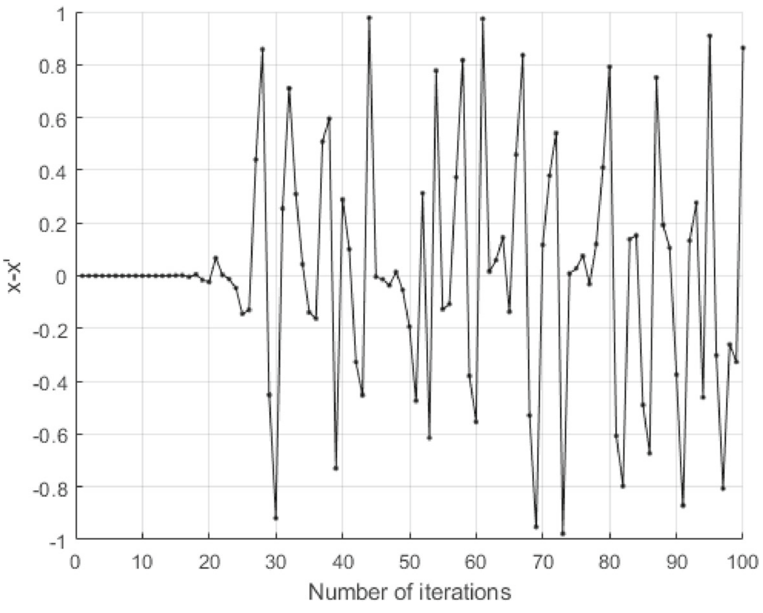


Fig. 2 Subtraction of two Logistic series

original decision boundary after embedding the watermark. We generate the dislocation sequence by Logistic chaos mapping to dislocate the original training set and give it the original label, which can achieve both of these requirements. After dislocation, the chunked histogram of the trigger set remains consistent with the original training set and is not correctly recognized by the neural network. When the model learns the trigger set features, the impact on the original classification accuracy of the model can be reduced because the label given to the trigger set by our method is the correct label of its original image. Our proposed neural network protection schemes to preprocess input images with a secret key before training or testing a model. A model f is trained by preprocessed images with a key K for the first time. To test the trained model, test images are also preprocessed with the same key K before testing so that the adversaries cannot use the model without the key. The next subsection details the preprocessing that is used in the proposed model protection method.

3 The details of our proposed method

To protect the intellectual property of neural network owners, we propose a novel watermarking protocol based on the Logistic chaotic mapping. We distribute the key through the authority. Then we use the key to generate two pairs of chaotic sequences to process the images to generate the trigger set. Finally, we put the original training set and the trigger set into the network together for training, so that the network learns both the classification task and the watermarking task, which means the embedding process of the watermark is completed. In the watermark verification phase, we submit the images to be processed (trigger set images before generation) to the validation authority, which processes the images according to the corresponding keys saved in the database and is authenticated by it without public disclosure to prevent attackers from more targeted attacks. Finally, when the network tag hit rate reaches a threshold value, that is, ownership verification is successful, otherwise, verification fails. The overall flow of the watermarking protocol is shown in Fig. 3.

The neural network protection method schemes to preprocess input images with a secret key before training or testing a model. A model f is trained by preprocessed images with a key K for the first time. To test the trained model, test images are also preprocessed with the same key K before testing so that the adversaries cannot use the model without the key.

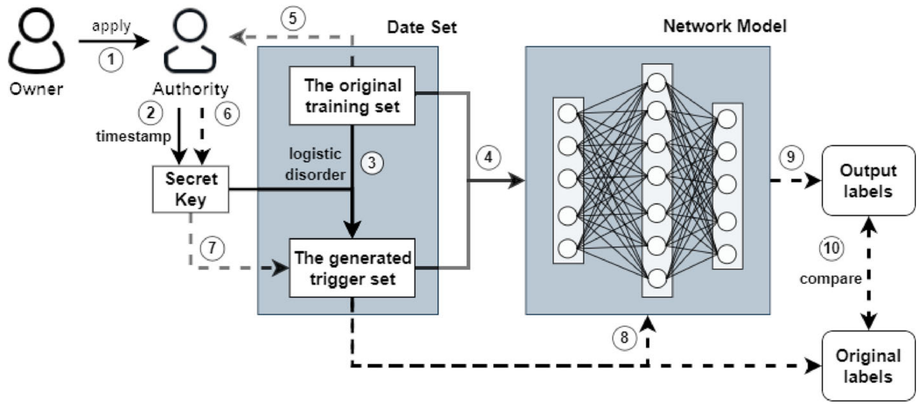


Fig. 3 Flow chart

]

The next subsection details the preprocessing that is used in the proposed model protection method.

3.1 Watermark generation

The process of watermark generation is steps ① to ③ in Fig. 3. We apply for a secret key K from the third-party authority, and the authority center generates the key K while recording the timestamp T. Then, we use the key K to generate two pairs of pseudo-random sequences for block randomization and intra-block randomization. The process of generating the scrambled sequences will be described in 3.4 Additional Content. The following are steps for preprocessing input images, where w and h denote the width and height of an image.

1. From left to right, top to bottom, divide x into the block with a size of M such that $\{B_1,B_2,\ldots,B_{\frac{w}{M}\times\frac{h}{M}}\}$
2. Generate a random sequence Q_1 and Q_2 using Logistic mapping
3. Sort the random sequence Q_1, Q_2 to generate the integer-disorder sequence A_1, A_2

$$A_1 = \{r_1, r_2, ..., r_u\} \tag{2}$$

   where r_i is the sorted value q_i^1 in the Q_1 sequence and $u = \frac{w}{M} \times \frac{h}{M}$; the r_i is a positive integer

$$A_2 = \{s_1, s_2, ..., s_v\} \tag{3}$$

   where s_i is the sorted value q_i^2 in the Q_2 sequence and $v = M \times M$; the s_i is a positive integer
4. Perform position replacement on each image block using A_1, and every new block B_i' as

$$B_i' = B_{r_i} \tag{4}$$

   where i indicates the position of the block, and r_i indicate the location of block replacement
5. The pixel values within each block are expanded in turn to obtain $\{B_i'(1), B_i'(2), ..., B_i'(M \times M)\}$. The pixel dislocation for each block using A_2 and generate a pre-processed image, and every pixel of each block

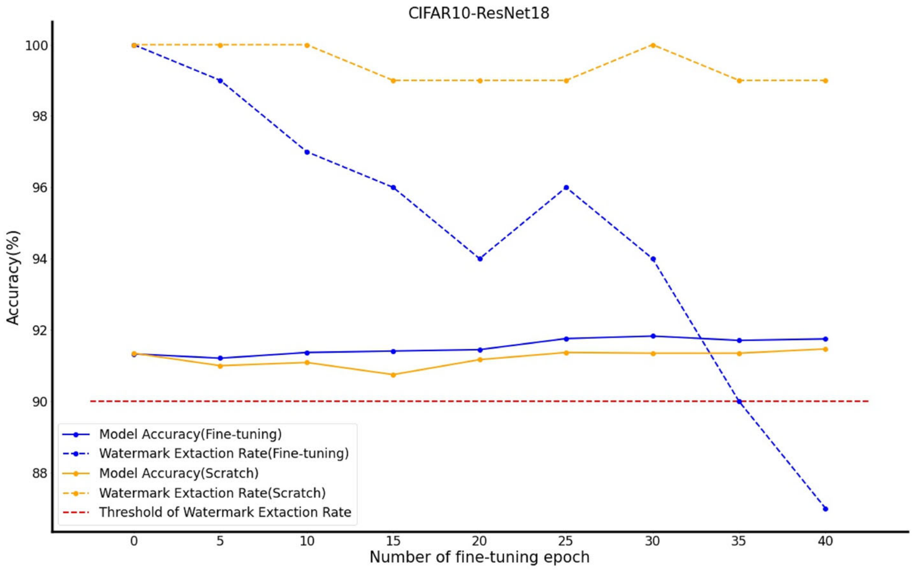
$$B_i''(j) = B_i'(s_j) \tag{5}$$

   where j indicates the position of each pixel in the block, and s_j indicate the location of pixel replacement in the block

Figure 4 is an example of a 32x32 pixel image used in our experiment. The small size of the image may result in some blurriness. At this point, we produce trigger set images as shown in Fig. 4 that have the characteristics that the unembedded neural network cannot classify correctly and retain some features of the original training set. We successfully embed the watermark while ensuring that the decision boundary of the original task is not affected and the watermark does not suffer from a false alarm rate. We summarize our concrete process of watermark generation in Algorithm 1.

3.2 Watermark embedding

The process of watermark embedding corresponds to step ④ in Fig. 3. In the embedding process of the watermark, we obtained from the entire training dataset $D = \{(x_1, y_1), ..., (x_m, y_m)\}$ where (x_i, y_i) is a pair of an input image and m is the size of the data set. We extract n images $(p_1, p_2, ..., p_n)$ and corresponding labels $(l_1, l_2, ..., l_n)$ from



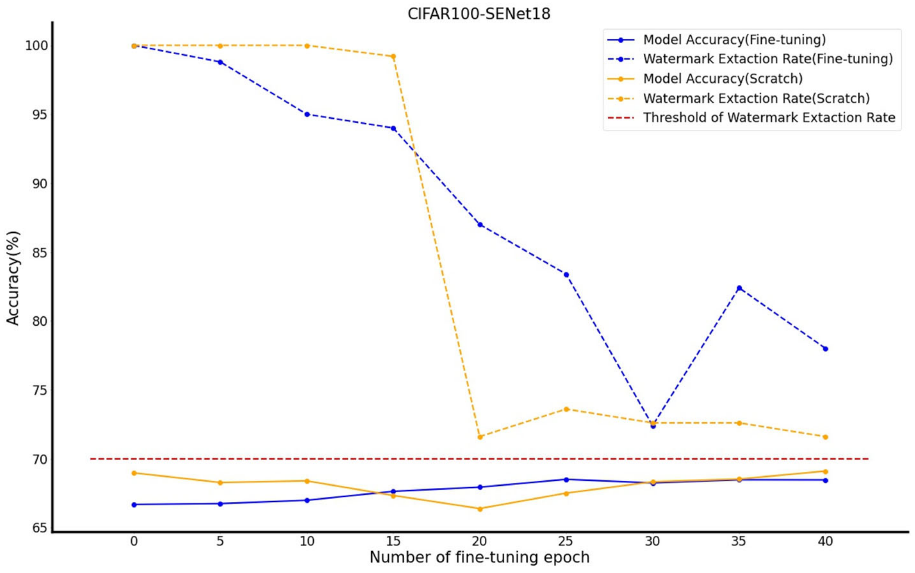


Fig. 4 Example of the trigger set

them, and then generate the trigger set $D_{wm} = \{(pe_1, l_1), ..., (pe_n, l_n)\}$ by the algorithm in 3.1 Watermark Generation. We put the original training set and the generated trigger set into the Model f training together, so that the neural network to be protected learns the mapping between the trigger set and the corresponding labels while learning the original classification task. Due to the special way our method constructs the trigger set, the trigger set remains consistent with the chunked mean histogram of the original training set images. This allows

Algorithm 1 Watermark Creating

```
Input:
        Number of trigger set: n
        Training set to be encrypted: $D_o = \{p_k, l_k\}_{k=1}^{n}$
        Block Randomization: $A_1$
        Intra-block Randomization: $A_2$
        Size of Block: M, M = 2,4,8...
Output:
        Trigger Set data: $D_{wm} = \{pe_k, l_k\}_{k=1}^{n}$

1: function Watermark_Creating()
2:
3: $D_{wm} \leftarrow \emptyset$
4: for each round i = 1,2,...n do
5:     $pe_i \leftarrow$ encrypt($p_i$)
6:     Add(($pe_i$, $l_i$), $D_{wm}$)
7: end for
8: return $D_{wm}$
```

the network to complete the embedding of the watermark in a relatively short time and with minimal impact on the original classification task of the network.

3.3 Watermark verifying

Algorithm 2 Watermark Verifying

```
Input:
        Training set to be encrypted: $D_o = \{p_k, l_k\}_{k=1}^{n}$
        DL Model h
        Number of trigger set: n
        Threshold $\tau$
Output:
        Owner's copyright verification results(True or False)

1: function Watermark_Verifying()
2:
3: correct $\leftarrow$ 0
4: for each round i = 1,2,...n do
5:     $pe_i \leftarrow$ encrypt($p_i$)
6:     $r_i \leftarrow$ f($pe_i$)
7:     if $l_i = r_i$ then
8:         correct $\leftarrow$ correct + 1
9:     end if
10: end for
11: if correct $> \tau$ then
12:     return True
13: else
14:     return False
15: end if
```

Finally, the verification phase of the watermark, as shown in steps ⑤ to ⑩ in Fig. 3. When a copyright dispute arises in our neural network, we can extract n images (p_1, p_2, ..., p_n) known only to ourselves and their original labels (l_1, l_2, ..., l_n) from the original training set, and then hand them over to authority center for authentication. The authority center obtains the trigger set $(pe_1, pe_2, ..., pe_n)$ by combining the images $(p_1, p_2, ..., p_n)$ using the same permutation algorithm with the key K kept in the database. Then, the authority center puts the trigger set into the network to be verified and derives t by comparing the consistency between the labels $(r_1, r_2, ..., r_n)$ output from the network and the real labels $(l_1, l_2, ..., l_n)$. When the calculated t is greater than the pre-defined threshold τ, in other words, $t \geq \tau$, ownership is verified. The specific authentication process can be seen in Algorithm 2.

3.4 Additionnal content

Since the current black-box watermarking approach in which the authority verifiers are only involved in the verification process is not sufficient to resist forgery attacks. We design a watermarking framework that is adapted to the trigger set construction method for the interaction between the owner and the authority center. The key for trigger set construction is kept by the authority, therefore avoiding the situation where an attacker forges his key to construct a trigger set.

For the generation process of two sets of pseudo-random permutation sequences, we first used Logistic chaos mapping for the initial values a_1, b_1 for multiple rounds.

$$a_n = \mu \cdot a_{n-1} \cdot (1 - a_{n-1}) \tag{6}$$

$$b_n = \mu \cdot b_{n-1} \cdot (1 - b_{n-1}) \tag{7}$$

where n is the number of iterations, $n \geq 100$

After a certain number of accumulations, new $q_1^1 = a_n$, $q_1^2 = b_n$ are obtained as the input values of Equation 1 to obtain two pseudo-random sequences $Q_1 = \{q_1^1, q_2^1, ..., q_n^1\}$, $Q_2 = \{q_1^2, q_2^2, ..., q_n^2\}$.

$$q_n^1 = \mu \cdot q_{n-1}^1 \cdot (1 - q_{n-1}^1) \tag{8}$$

where n is the number of image blocks

$$q_n^2 = \mu \cdot q_{n-1}^2 \cdot (1 - q_{n-1}^2) \tag{9}$$

where m is the number of pixel points in the image block

4 Experiments

In this section, we evaluate the performance of our watermarking protocol on two training sets on two different neural networks. First, we will introduce some relevant settings for our experiments. Then we will test the fidelity and effectiveness of watermarking experiments. Finally, we will test the robustness of watermark under various attacks (fine-tuning attack, compression attack, overwriting attack).

4.1 Experimental settings

We train and test the ResNet18 [7] and SENet18 [8] using two image classification data sets: CIFAR10 and CIFAR100, for each network. CIFAR10 and CIFAR100 are two renowned datasets widely employed in the field of machine learning and computer vision for image classification tasks. CIFAR10 comprises 60,000 color images with a dimension of 32x32

Table 1 Effectiveness analysis

Model	Data set	Unmarked Model Watermark Extraction Accuracy	Watermark Extraction Accuracy	
			Fine-tuning	Scratch
ResNet18	CIFAR10	13.00	100.00	100.00
SENet18	CIFAR100	0.40	100.00	100.00

pixels, categorized into 10 classes, each containing 6,000 images. The classes include common objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. This dataset is highly regarded as a benchmark for assessing image classification models due to its small size, simple structure, and high-quality annotations. On the other hand, CIFAR100 poses a greater challenge for image classification, with 100 classes, each containing 600 images. The classes are divided into 20 superclasses, with five subclasses in each. This dataset offers a more diverse set of images, including insects, flowers, household appliances, and vehicles, providing a broader range of objects for model training and evaluation. In [2, 27] the trigger set is generated by giving "error labels" to the original images, so it distorts the decision boundaries of the original network. Our approach is to generate a scrambled sequence by an irreversible encryption algorithm, which encrypts the images and assigns their original labels while keeping the image chunking mean histogram unchanged, thus reducing the impact on the original classification task of the network.

4.2 Effectiveness and false alarm rate

The effectiveness of watermark refers to the ability of neural network owners to successfully verify their copyright in our Logistic chaos encryption algorithm scheme. It requires that the model with an embedded watermark has a very high accuracy rate for the input trigger set verification.

In contrast, FAR analysis requires a low hit rate for the input trigger set of the network without an embedded watermark to ensure that the model does not have problems such as false positives. Table 1 shows the watermark accuracy of the private model before and after embedding the watermark respectively.

From Table 1, we can see that the extraction rate of the model for the trigger set is only 13% and 0.4% when the model is not trained with watermark information. This corresponds to a random probability for a ten-class task, which means that the model cannot recognize the trigger set. After the model is trained on the trigger set, the extraction rate of the embedded watermark can reach 100%, either by fine-tuning the training of the embedded watermark or by training from scratch, which indicates that the total watermark information embedded can be correctly recognized and proves the validity of the watermark.

4.3 Fidelity

The fidelity experiments are designed to test the magnitude of the impact of watermark embedding on the original performance of the protected model.

In Table 2, the values in the third column indicate the classification accuracy of the original task when the model is not embedded in the watermark, while the accuracy of the

Table 2 Fidelity analysis

Model	Data set	Unmarked Model Accuracy	Marked Model Accuracy	
			Fine-tuning	Scratch
ResNet18	CIFAR10	91.38	91.33	91.35
SENet18	CIFAR100	68.03	66.68	68.97

original classification task after the model is embedded in the watermark is indicated in the fourth and fifth columns. We can see that for the model ResNet watermark embedding degrades by only 0.05 percentage points when using fine-tuning embedding and by only 0.03 percentage points when embedding the watermark using the scratch approach. In model SENet, the classification accuracy decreases by 1.35 percentage points when embedding the watermark by fine-tuning, but improves by 0.94 percentage points when embedding the watermark by scratch. As shown in Table 2, the model does not change significantly in model accuracy before and after watermark embedding, which also indicates that our watermarking framework distorts the original classification decision boundary of the model to a negligible extent.

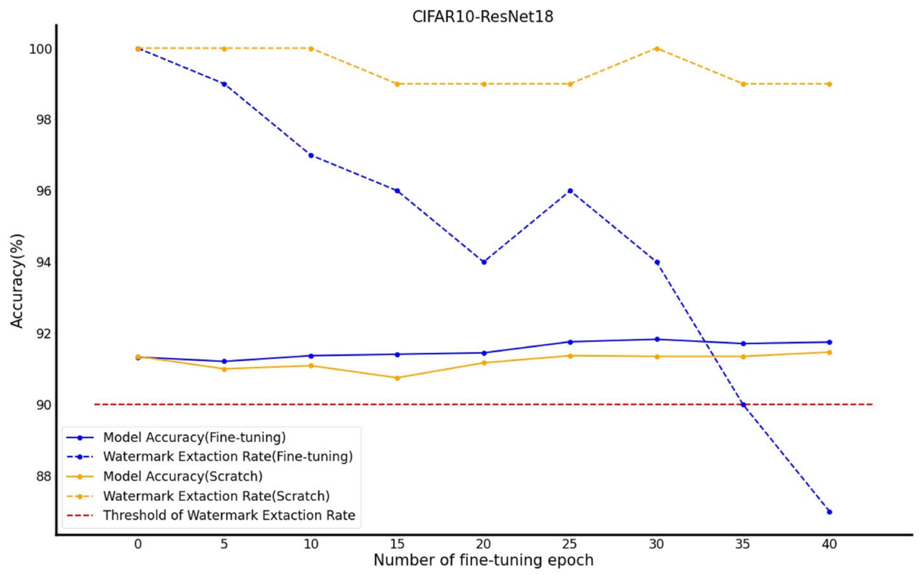
4.4 Robustness analysis

Robustness is one of the key elements for the neural network watermark. Unlike normal digital watermarks, neural network models black-box watermarking may occur with verification failure after fine-tuning and retraining of the network. And the embedded watermark also needs to be resistant to attackers using network cropping, watermark overwriting, and other ways to attack the neural network. In the following subsections, we assess the robustness of our model against three types of attacks: fine-tuning, model compression and watermark overwriting.

4.4.1 Model fine-tuning

Model fine-tuning refers to users of the network retraining the neural network model with a certain amount of training data. As the model decision boundaries are slightly altered, this may result in the corruption or the disappearance of the watermarking information in the model. In our experiments, we assume that the user gets the network model and then trains it for 40 epochs of fine-tuning to adjust the network. We determine whether the watermark can resist the fine-tuning attack by verifying whether the hit rate of the trigger set is higher than the threshold τ. Here we set the threshold value of 90% for the CIFAR10 dataset corresponding to the trigger set and 70% for the CIFAR100 dataset corresponding to the trigger set.

In Fig. 5 we record the original accuracy of the model and the verification accuracy of the watermark after fine-tuning 0 - 40 epochs for the two networks with different embedding methods. In Fig. 5, the solid line shows the original classification accuracy of the model, and the dashed line shows the extraction accuracy of the model for the watermark. The blue and orange colors represent two different watermark embedding methods: fine-tuning and scratch. We can obtain that the watermarking accuracy can be maintained above the threshold in all cases except when the watermark is embedded in the ResNet network using the fine-tuning approach, where the watermarking accuracy drops to 87% after 40 epochs



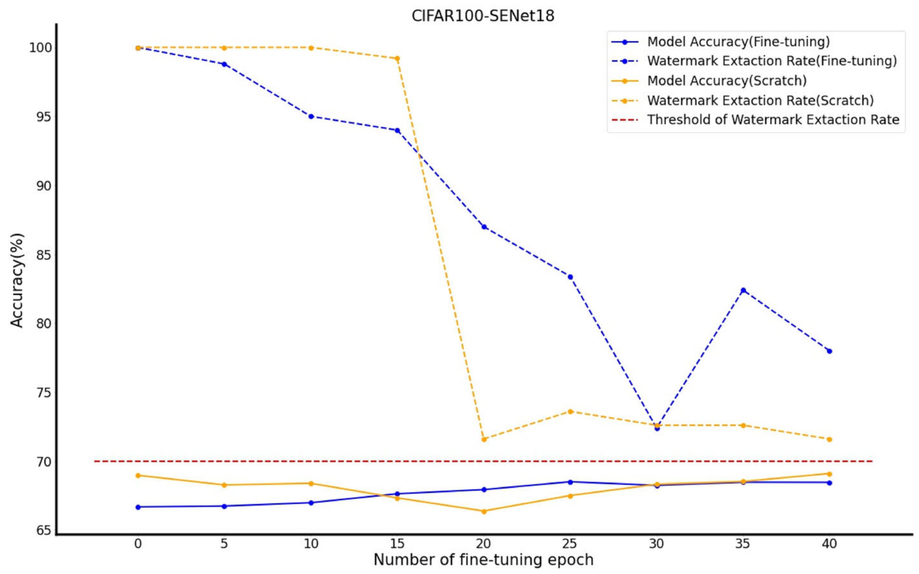
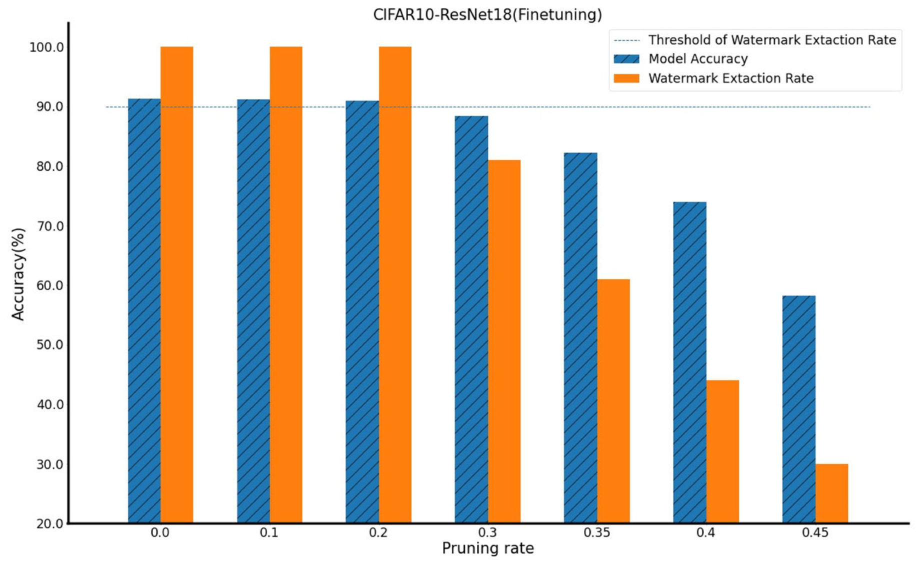


Fig. 5 Robustness against fine-tuning attacks

of fine-tuning training. This also indicates that our watermark is highly robust to fine-tuning training.

4.4.2 Model compression attacks

To further compress deep learning models, while ensuring that the models have the same accuracy as before, model compression methods are becoming increasingly useful in deep



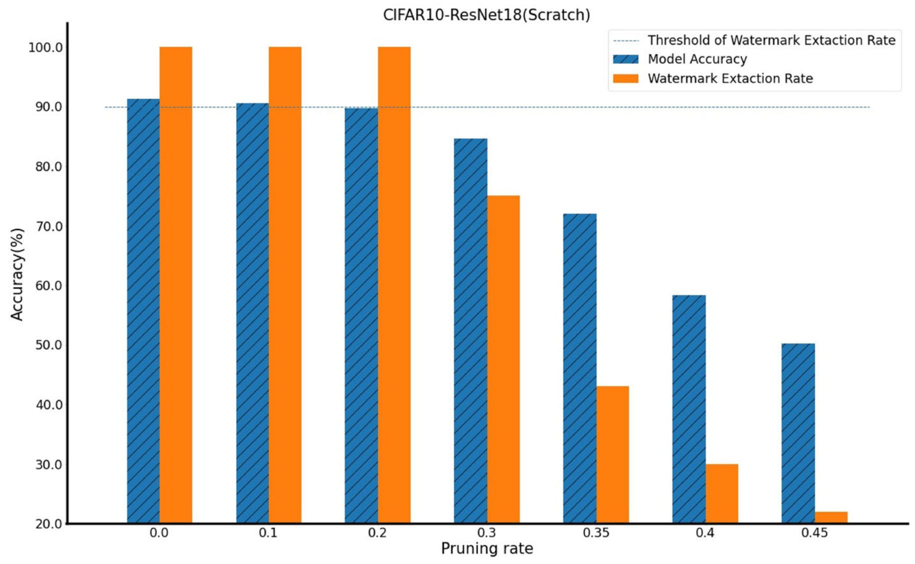
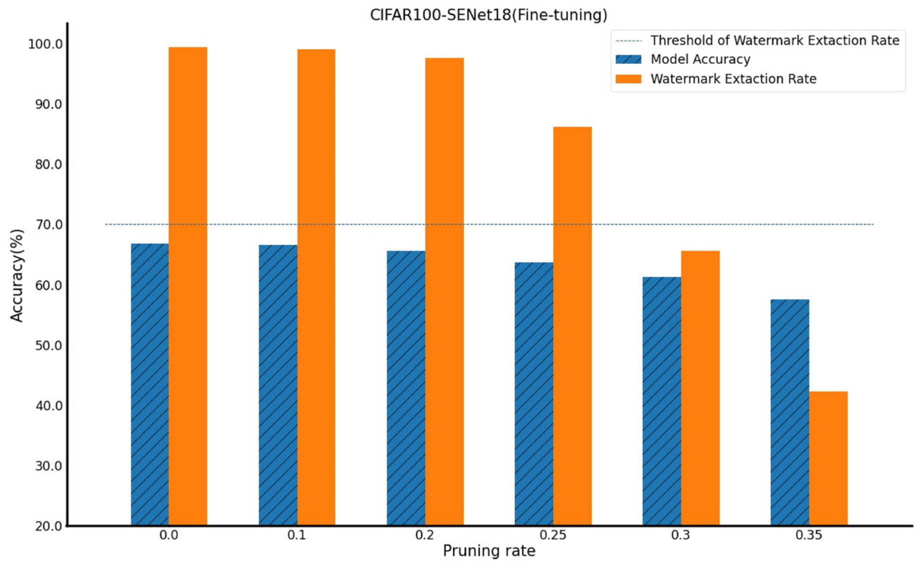


Fig. 6 Robustness against compression attacks (ResNet18)

learning models. There examining the ability to resist compression attacks is also part of verifying the robustness of the watermark. We use the TensorFlow Model Optimization Toolkit to prune our model. Figures 6 and 7 show the effect of model compression on watermark detection accuracy and original task accuracy at different trimming rates.

In Figs. 6 and 7, the blue column indicates the accuracy of the model's original classification task, and the orange column indicates the watermark detection accuracy. The blue dashed line indicates the threshold value that needs to be reached for the accuracy of the



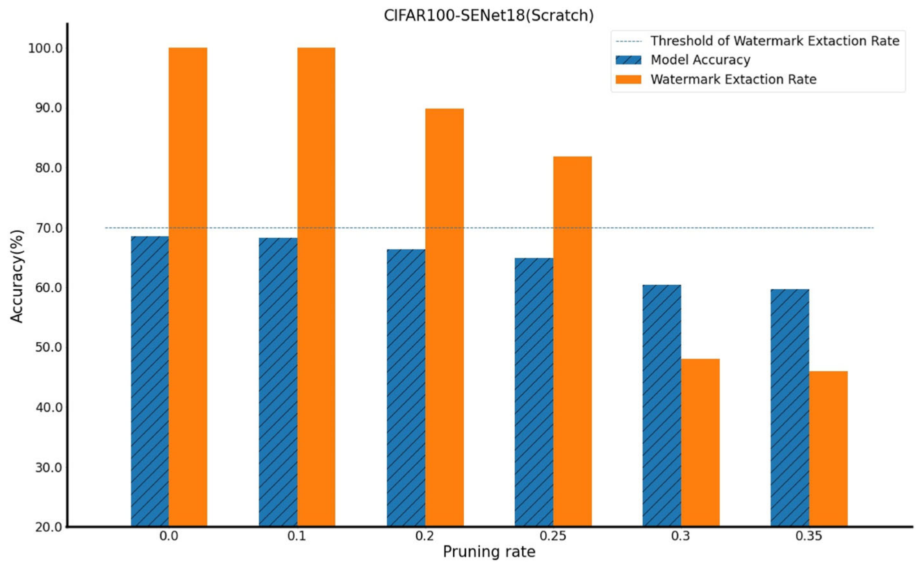


Fig. 7 Robustness against compression attacks (SENet18)

model watermark. The x-axis indicates the ratio of model pruning, and the y-axis indicates classification accuracy. The first of the bar charts in Fig. 6 shows the values for the model with the fine-tuning method of embedding the watermark, the second bar chart shows the values for the model with the scratch method of embedding the watermark, and Fig. 7 is based on the same principle. Figure 6 shows before the pruning rate are less than 25%, the detection accuracy of the watermark is greater than the threshold value of 90%. When the pruning rate exceeds 25%, the original classification accuracy of the model is also greatly reduced, which

Table 3 Robustness against overwriting attacks (ResNet18)

Owner's type of watermark embedding method	Fine-tuning			Scratch		
Date ratio (%)	10	20	30	10	20	30
Original model accuracy (%)	91.33	91.33	91.33	91.26	91.26	91.26
Model accuracy after training (%)	87.20	90.25	91.66	87.20	90.45	91.16
Accuracy of owner's WM extraction after training (%)	45.00	63.00	89.00	32.00	58.00	91.00
Accuracy of attacker's WM extraction after training (%)	84.00	100.00	100.00	100.00	100.00	100.00

means that the model has lost its original function and the validation is meaningless. In Fig. 7 for the SENet18 network experiments, we set the watermarking accuracy threshold to 80% for one hundred classifications. We can see in the figure that the watermarking accuracy can be kept above the threshold when the model pruning rate is less than 25%. While the pruning rate exceeds 25% the original accuracy of the model decreases seriously and the model has lost its original value.

4.4.3 Watermark overwriting attacks

One effective way for the attacker to cause a copyright dispute over a neural network model is an overlay attack. In our experiments, we assume that the attacker knows the watermarking construction method, but only knows at most 30% of the original training data, and does not know our key sequence. We assume that the attacker generates the trigger set by his key and retrains the network. Finally, we count the impact of the attacker on the original watermark with 10%, 20%, and 30% of the training data.

As can be seen in Table 3, the attacker's watermark is indeed successfully embedded and the owner's watermark is partially erased in the cases where the attacker owns 10% and 20% of the dataset. But the original accuracy of the model is much degraded, which the attackers do not want to get. And in the case where the attacker has 30% of the dataset, although the model accuracy remains the same, the original watermark of the owner is not well erased, which also indicates that the attacker's overwriting attack fails.

Table 4 Robustness against overwriting attacks (SENet18)

Owner's type of watermark embedding method	Fine-tuning			Scratch		
Date ratio (%)	10	20	30	10	20	30
Original model accuracy (%)	66.68	66.68	66.68	68.97	68.97	68.97
Model accuracy after training (%)	62.90	65.80	65.56	57.00	66.10	66.70
Accuracy of owner's WM extraction after training (%)	5.40	32.20	37.00	3.40	22.00	45.80
Accuracy of attacker's WM extraction after training (%)	13.20	39.60	95.80	13.40	100.00	100.00

Table 5 Model fidelity of Zhang et al. [27], Zhong et al. [28], Maung and Kiya [16] and our method

Model	Unmarked Model Accuracy(A_0)	Marked Model Accuracy(A_1)	fidelity($\frac{A_0-A_1}{A_0} \times 10^3$)
Zhang et al.	78.6	78.49	1.40
Zhong et al.	91.92	87.46	48.52
Maung et al.	95.45	92.99	25.77
Proposed	91.38	91.35	0.33

Table 4 summarizes that for the SENet18 network whether the attacker has 10%, 20% or 30% of the dataset overlaying the watermark leads to severe degradation of the model accuracy, which also shows that our watermark is strongly robust to overwriting attacks.

4.5 Comparison with existing

Since the current black-box watermarking methods embed different models, we use a more reasonable way to compare the model fidelity. The specific calculation formula is as follows:

$$f = \frac{A_0 - A_1}{A_0} \times 10^3 \tag{10}$$

Where f is the influence of watermark embedding on model fidelity, A_0 is the unmarked Model Accuracy, A_1 is the marked Model Accuracy. The smaller the value of f, the lower the effect of watermark embedding on the original accuracy of the model.

In the experiment, we use the CIFAR10 dataset as a general dataset and compare it with literature [16, 27, 28]. After generating the corresponding watermark samples, we use 50 epochs of fine-tuning to train the network to embed the watermark.

Table 5 displays three columns of data. The first column represents the model's classification accuracy prior to watermark embedding, while the second column represents the accuracy post-embedding. Finally, the last column demonstrates the extent of the proposed watermark protocol's impact by showcasing the original accuracy calculated using Equation 10. Tables 5 demonstrate that the approaches proposed by Zhong et al. [28] and Maung and Kiya [16] have a more significant impact on the accuracy of the model, leading to a reduction in the original classification accuracy. This outcome contradicts our primary objective of safeguarding the model. While Zhang et al. [27] method has a less severe impact on the model's accuracy, our proposed approach still outperforms it, offering enhanced performance and resistance against potential forgery attacks by adversaries.

5 Conclusion

We propose a novel watermarking protocol to protect the intellectual property of neural networks. We use a Logistic chaos mapping-based dislocation approach to generate trigger sets, which ensures that the model cannot correctly classify the trigger sets when the watermark is not embedded. And the trigger set retains the histogram and labels of the original images unchanged, which makes the model less influential on the original decision boundary when learning the watermarking task. Because of features such as Logistic nonlinearity and unpredictability, an attacker cannot predict the permutation sequence. In the watermarking verification process, the entire verification process of the authority is not open to the public, which also makes it impossible for attackers to find the features of our trigger set, thus effectively resisting the more targeted forgery attacks and overwriting attacks. This paper proposes a watermarking framework for achieving copyright protection of deep neural models. However, this is just the beginning. The protection of intellectual property for deep neural networks is a relatively new problem, and there is still much room for improvement and further research. Currently, most of the proposed watermarking frameworks for protecting the copyright of deep neural network models are based on the classification tasks of neural network models. Designing a watermarking framework that is applicable to any type of neural network model will have broad prospects. Therefore, as future work, we plan to continue researching methods to make watermarks more resistant to stronger and more targeted attacks, and to explore different watermark embedding techniques to adapt to different types of neural network models.

Acknowledgements It is an honor to be part of Dr. Shen's team. I would also like to thank my partner for her great support in my work. The Basic Research partially supported the project (Grant No.2020B1515120089, No.2021A1515011171, No.202102080410, and No.202102080282).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Adi Y, Baum C, Cisse M, Pinkas B, Keshet J (2018) Turning your weakness into a strength: watermarking deep neural networks by backdooring. In: 27th {USENIX} Security Symposium ({USENIX} Security 18). pp 1615–1631
2. Asikuzzaman M, Pickering MR (2017) An overview of digital video watermarking. IEEE Trans Circuits Syst Video Technol 28(9):2131–2153
3. Dargan S, Kumar M, Ayyagari MR, Kumar G (2020) A survey of deep learning and its applications: a new paradigm to machine learning. Arch Comput Meth Eng 27(4):1071–1092
4. Darvish Rouhani B, Chen H, Koushanfar F (2019) Deepsigns: an end-to-end watermarking framework for ownership protection of deep neural networks. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. pp 485–497
5. Deng L, Yu D (2014) Deep learning: methods and applications. Found Trends Signal Process 7(3–4):197–387
6. Guo J, Potkonjak M (2018) Watermarking deep neural networks for embedded systems. In: 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, pp 1–8
7. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp 770–778
8. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp 7132–7141
9. Jia H, Choquette-Choo CA, Chandrasekaran V, Papernot N (2021) Entangled watermarks as a defense against model extraction. In: 30th {USENIX} Security Symposium ({USENIX} Security 21)
10. Le Merrer E, Perez P, Trédan G (2020) Adversarial frontier stitching for remote neural network watermarking. Neural Comput Appl 32(13):9233–9244
11. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
12. Li Z, Hu C, Zhang Y, Guo S (2019) How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp 126–137
13. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. Springer, pp 740–755

14. Liu Y, Tang S, Liu R, Zhang L, Ma Z (2018) Secure and robust digital image watermarking scheme using logistic and RSA encryption. Expert Syst Appl 97:95–105
15. Li H, Wenger E, Shan S, Zhao BY, Zheng H (2019) Piracy resistant watermarks for deep neural networks. Preprint at http://arxiv.org/abs/1910.01226
16. Maung Maung AP, Kiya H (2021) Piracy-resistant DNN watermarking by block-wise image transformation with secret key. In: Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security. pp 159–164
17. May RM (2004) Simple mathematical models with very complicated dynamics. The Theory of Chaotic Attractors 85–93
18. Namba R, Sakuma J (2019) Robust watermarking of neural network with exponential weighting. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp 228–240
19. Shafer DS (1995) Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering (Steven H. Strogatz). SIAM Rev 37(2):280–281
20. Szyller S, Atli BG, Marchal S, Asokan N (2021) Dawn: dynamic adversarial watermarking of neural networks. In: Proceedings of the 29th ACM International Conference on Multimedia. pp 4417–4425
21. Uchida Y, Nagai Y, Sakazawa S, Satoh S (2017) Embedding watermarks into deep neural networks. In: Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval. pp 269–277
22. Wang J, Wu H, Zhang X, Yao Y (2020) Watermarking in deep neural networks via error back-propagation. Electron Imag 2020(4):22–1
23. Wang T, Kerschbaum F (2019) Attacks on digital watermarks for deep neural networks. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp 2622–2626
24. Wang T, Kerschbaum F (2021) Riga: covert and robust white-box watermarking of deep neural networks. In: Proceedings of the Web Conference 2021. pp 993–1004
25. Yang F, Mou J, Liu J, Ma C, Yan H (2020) Characteristic analysis of the fractional-order hyperchaotic complex system and its image encryption application. Signal Process 169:107373
26. Zhang LY, Zheng Y, Weng J, Wang C, Shan Z, Ren K (2018) You can access but you cannot leak: defending against illegal content redistribution in encrypted cloud media center. IEEE Trans Dependable Secure Comput 17(6):1218–1231
27. Zhang J, Gu Z, Jang J, Wu H, Stoecklin MP, Huang H, Molloy I (2018) Protecting intellectual property of deep neural networks with watermarking. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. pp 159–172
28. Zhong Q, Zhang LY, Zhang J, Gao L, Xiang Y (2020) Protecting IP of deep neural networks with watermarking: a new label helps. Advances in Knowledge Discovery and Data Mining 12085:462