Journal of Computers Vol. 35 No. 4, August 2024, pp. 277-290
doi: 10.53106/199115992024083504019

Black Box Watermarking for DNN Model Integrity Detection Using Label Loss

Yunfei Song[1], Yujia Zhu[1], Liu Yang[1],
and Daoxun Xia[1,2,3*]

[1] School of Big Data and Computer Science, Guizhou Normal University,
Guiyang, 550000, China
{1287144516, 1183402473, 1437904070}@qq.com

[2] Guizhou Key Laboratory of Information and Computing Science, Guizhou Normal University,
Guiyang, 550000, China
dxxia@gznu.edu.cn

[3] Engineering Laboratory for Applied Technology of Big Data in Education, Guizhou Normal University,
Guiyang, 550000, China

Received 10 November 2023; Revised 4 April 2024; Accepted 25 April 2024

Abstract. After significant investments of time and resources, the accuracy of deep neural network (DNN) models has reached commercially viable levels, leading to their increasing deployment on cloud platforms for commercial services. However, ongoing research indicates that the challenges facing deep neural network models are continually evolving, particularly with various attacks emerging to compromise their integrity. Deep neural networks are susceptible to poisoning attacks and backdoor attacks, both of which involve malicious fine-tuning of the deep models. Malicious fine-tuning can lead to unpredictable outputs from deep neural network models. Although at-tempts have been made to address this issue, these solutions often increase model complexity or diminish model performance. We propose a black-box watermarking technique based on trigger image sets, which can effectively detect malicious fine-tuning while also enabling copyright authentication. This watermarking technique builds upon black-box watermarking methods, leveraging trig-ger image sets and utilizing a two-stage alternating training approach to fine-tune the model. During training, a novel loss function is employed to optimize the trigger images, thereby embedding the watermark while preserving the model's original classification capabilities. The proposed watermarking model is highly sensitive to malicious fine-tuning, resulting in unstable classification outcomes for trigger images. Ultimately, by inputting trigger image sets and analyzing the output of the watermarking model, the integrity of the deep neural network model can be verified. Experimental results demonstrate the effectiveness of this approach in detecting the integrity of DNN models.

Keywords: deep neural network, watermarking, trigger set, copyright protection

1 Introduction

With the development of computer technology, Deep Learning (DL) has made tremendous progress in various fields such as computer vision [1], semantic recognition [2, 3] and natural language processing [4]. After significant investments of time and resources, the accuracy of Deep Neural Network (DNN) models has now reached commercially viable levels, leading to their increasing deployment on cloud platforms for business services. However, ongoing research indicates that the challenges facing deep neural network models are continually evolving, particularly with various attacks emerging to compromise their integrity. This paper presents a novel black-box watermarking technique based on trigger image sets, which can effectively detect whether a deep neural network (DNN) model has been fine-tuned and ensure the fidelity of the model's performance. A set of trigger images is constructed based on user-specific keys, and the pre-trained DNN model is fine-tuned by alternately classifying normal images and trigger images to embed the watermark. In the verification stage, the integrity of the model is validated by analyzing the classification results of the model with the trigger image set as input. During the model embedding stage, both the generative network and classification network are simultaneously

* Corresponding Author

employed to alternately train the trigger image set, thus enhancing the accuracy of black-box trigger set detection. Compared to other black-box watermarking methods, this approach provides a new and effective solution for verifying the integrity of DNN models under commercial black-box systems.

Cybersecurity techniques are also evolving with the development of DNN models, but so are the attack methods against the models. DNN models besides face model modification attack methods such as fine-tuning [5], pruning, compression and model retraining. They are also vulnerable to data poisoning and backdoor attacks [6]. Data poisoning [7] reduces the accuracy of the model by contaminating the training data to bias the model's classification of good and bad inputs. Backdoor attacks achieve misdirection of model classification through secret triggers. Both attacks involve malicious fine-tuning of DNN models. And this kind of fine-tuning is more insidious than model modification and difficult to detect directly.

After analyzing existing literature, it has been found that deep neural networks are susceptible to data poisoning and backdoor attacks. Both involving malicious fine-tuning of the model. To date, most published studies have focused on robust watermarking techniques for protecting the copyrights of DNN models. Robustness implies that these methods are insensitive to changes aimed at removing embedded watermarks. In summary, detecting whether a deep neural network model has undergone fine-tuning is the key problem addressed in this paper. We aim to achieve both verification and copyright authentication. We will transplant the concept of sensitivity to content modifications into deep neural networks and integrate black-box watermarking techniques to address the aforementioned issues.

Black-box watermarks rely solely on input-output analysis. However, the integrity of the model greatly influences the watermark's output. Particularly, when the model faces attacks like fine-tuning or tampering by attackers, it may perform similarly to the original model. Yet, it could output different results when given specific backdoor data, hindering model validation. Meanwhile, we found that the training trigger set can also have an impact on the DNN model. Drawing inspiration from fragile watermarking, we contemplate the feasibility of utilizing black-box watermarking as a foundation. This entails analyzing the outcomes of the trigger image set to ascertain whether the model has undergone fine-tuning. The contributions of this paper include the following:

1) We present a novel black-box scheme for detecting model integrity using trigger set images, offering a fresh and efficient approach for verifying the integrity of DNN models within commercial black-box systems.
2) The embedding stage of the model utilizes both the generative network and classification network concurrently, alternating the training of trigger set images. This approach enhances the accuracy of detecting trigger sets in black-box scenarios.
3) The watermarking scheme adeptly identifies whether the DNN model has undergone fine-tuning, ensuring the fidelity of its performance.

2 Related Work

This section gives a detailed introduction to the work related to parameter-based watermarking and backdoor-based watermarking. Parameter-based watermarking enhances comprehension of the structure of DNN models. On the other hand, backdoor-based watermarking delves into the evolutionary history of black-box watermarking, alongside the contributions made by previous researchers. The approach used in this paper involves the use of sample adjustment to achieve a black-box model. So this section gives an introduction to dataset adjustment as well, describing the development of having an impact on the model by adjusting the dataset and the related work that has been done.

2.1 Parameter-based Watermarking

DNN models have a large number of parameters in them. These parameters can store a large amount of information other than the main task and can be utilized for DNN model watermark embedding [8]. Y. Uchida et al. [9] proposed the usage of watermarks for the first time in the DNN model to protect DNN copyright, and implemented an algorithm to embed watermark information in the model parameter weights during the training process. Kuribayashi et al. [10] proposed a quantifiable watermark embedding method based on the weights of the fully connected layer. This method exploits the fact that the weight of the DNN model varies greatly when parameter-based watermarking is added to the model. The existence of watermarks can be detected by analyzing the

change in weight. B. Cortinas-Lorenzo et al. [11] also found that the optimization algorithm of the deep model affects the validity of the watermarking information of the DNN model. The Adam optimization algorithm leads to significant changes in the distribution of the weight parameters of the DNN model, and the watermarking information is more easily detected. They proposed the Adam optimization algorithm based on orthogonal block projection, which optimizes the weight of the DNN model without changing the model to avoid watermark information being detected by attackers.

Different from the traditional way of modifying parameters to realize watermark embedding. Key-based model protection schemes have been proposed. Xue et al. [12] proposed an active intellectual property protection technique called "ActiveGuard" for deep neural networks. This technique leverages adversarial examples as users' fingerprints to ensure the security and protection of the model's intellectual property. By introducing a control layer into the model and using adversarial examples as keys, authorized access to the model is achieved. While Fan et al. [13] add a passport layer after each convolutional layer. If the user does not have the correct key to pass through the passport layer, the performance of the model decreases drastically. Tian et al. [14] used a hierarchical access approach. Encryption algorithms are utilized to encrypt important parameters in the DNN model. Different levels of access can decrypt different number of parameters. Zhao et al. [15] proposed the copyright protection method of DNN model by using the pruning rate as the key, which only provides the correct pruning rate for authorized users. An unauthorized user cannot recover the full performance of the DNN model without the correct key.

2.2 Backdoor-based Watermarking

Backdoor attack [16] is a type of attack on the DNN model. The attacker trains the model and the model input a specific label when a specific input appears. A particular input is defined as one or more instances in the deep model, and the set of these instances is called the trigger set. The DNN model can be trained with the function of the classification trigger set by taking advantage of the overparameterized characteristic of the DNN model. Adding backdoors to the model does not degrade the performance of the model on the original task [17]. Yossi et al. [18] used the backdoor as the watermark key image and took advantage of the overparameterized feature of DNN. In this scheme, the corresponding label of the trigger set is randomly selected. Finally, watermark detection is realized by comparing the accuracy and threshold of the trigger set. Li et al. [19] proposed a copyright protection framework for the DNN model based on blind watermarking. Through this framework, the identity of the owner of the model and the DNN model is associated, and the watermark key trigger set is generated by taking the ordinary trigger set and the specific trigger set as the input.

Wu et al. [20] trained the host network and watermark extraction network together, and the trained neural network also embedded watermarks in the output image while realizing the original task. By detecting the watermark in the output image, the copy-right ownership of the host network is identified, and whether the image comes from a neural network is determined. Linna Wang et al. [21] proposed a dynamic watermarking method based on a reversible image hiding network, which enhances the watermark's imperceptibility and accuracy.

The advantages and disadvantages of these methods are as follows: The backdoor attack method is simple and easy to implement, allowing for the modification of model behavior without detection. However, it is prone to detection and loses effectiveness once discovered. Watermark-based methods can associate the model with its owner, thus protecting intellectual property. However, they may impact model performance and could become ineffective, especially under adversarial attacks. Training both the host network and watermark extraction network simultaneously enables watermark embedding while maintaining model performance, but it requires more complex training and design, potentially increasing computational costs. Dynamic watermarking methods can enhance the concealment and accuracy of watermarks but may introduce additional complexity and computational overhead. Upon analyzing the aforementioned methods, we have observed that the majority of research focuses on robust watermarking techniques aimed at protecting the copyright of DNN models, which exhibit insensitivity to variations in the embedded watermark. Therefore, the key issue addressed in this paper is to utilize black-box watermarking techniques to detect whether deep neural networks have been subjected to malicious fine-tuning. Additionally, the watermarking method should also enable copyright authentication. Thus, to address both the model integrity verification and copyright ownership verification issues, we propose the approach presented in this paper.

We combine the concept of traditional fragile watermarking with black-box watermarking techniques for watermarking deep neural network models. For the application of this approach, we need to consider the following issues. Firstly, it should be easy to extract from and embed into the model, with low training costs. Secondly, the

embedded watermark should be imperceptible and have minimal impact on the original neural network. Lastly, there should be quantifiable metrics to address malicious fine-tuning, and the watermark should also be scalable.

2.3 Dataset Adjustment

Datasets have a strong correlation with deep neural network models, and the structure of the neural network models can affect the results of dataset classification. Y. Adi et al. [18] proposed constructing a trigger set by using a set of abstract images and tags that are not related to the image content. The tags are generated randomly and are independent of each other. Datasets can also deceive deep neural network models. N. Carlini et al. [22] proposed that adversarial examples (AEs) be used to confuse neural networks. Adversarial examples are indistinguishable to the human eye, but can interfere with neural networks to produce unreliable decisions. Labels that trigger misallocation of datasets more or less distort the original decision boundary. Q. Zhong et al. [23] minimizes (or even eliminates) the effect of distorting he original decision boundary by adding new class tags to the well-crafted key samples during training. Consequently, the protection of the neural network model can also be considered from a dataset perspective, by generating new trigger datasets and modifying them. The review written by Linna Wang et al. [24] also mentioned the impact of datasets on DNN models in dynamic watermarking, and introduced the authenticity of datasets when used for copyright authentication.

3 Methodology

We propose a black-box watermarking framework for DNN models. The core operation process of the framework is shown in Fig. 1. Trigger images are trained cyclically in the generative network and the target network to finally realize the DNN model watermark embedding. In this section we describe in detail the training process of watermark embedding, the loss error of model watermarking during training, and the method of model validation proposed in this section.

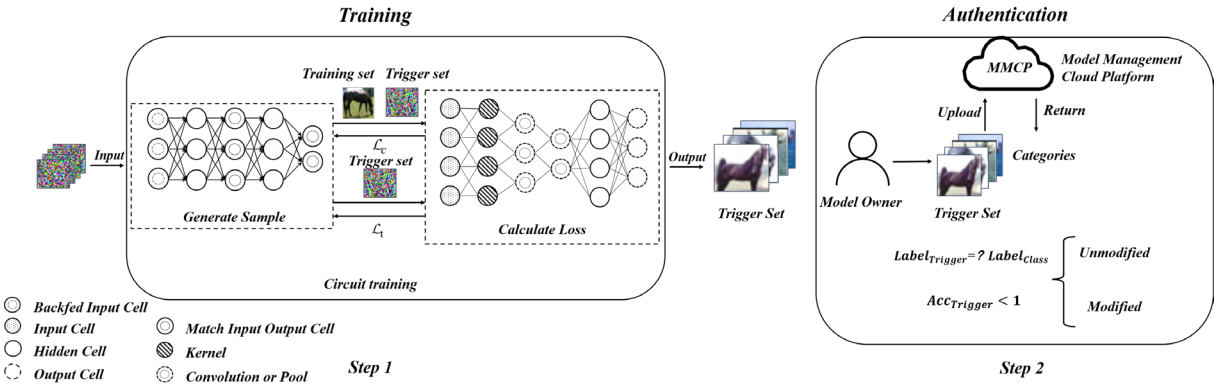

Fig. 1. The overall flow of the proposed

(Our watermarking scheme consists of two parts: training and authentication. The training part represents the three stages of trigger image set generation, trigger image set classification training and model watermark embedding (see Section 3.2). The authentication part rates whether the model has been modified by comparing the classification labels with the original labels of the trigger images and measuring the trigger set accuracy (see Section 3.4).)

3.1 Overview

This chapter will provide a detailed overview of the design process of the black-box watermarking technique based on trigger image sets. The proposed black-box watermarking method can not only protect deep neural models but also verify the integrity of the models. The introduction elaborates on the background and signif-

icance of the design. Subsequently, the chapter delves into the design process of the watermarking scheme, including the trigger image training phase, watermark embedding process, and watermark verification process. When discussing watermark training losses, the chapter will focus on analyzing losses during the trigger image training phase and the watermark embedding phase. Next, experimental results and analysis will be presented, including an introduction to the experimental dataset, the impact of different training set selections, model watermark performance, comparisons with existing methods, and model attack detection. Finally, a summary of the chapter's content will be provided, laying the groundwork for the discussion in subsequent chapters.

3.2 Watermark Training

Fig. 1 illustrates the overall architecture diagram of the watermarking proposed in this paper. We focus on the model watermark training part in this section. The process consists of three phases: generation of trigger image set, classification training of trigger image set and model watermark embedding.

Trigger Image Set Generation Phase. The trigger image set is related to the DNN model user and a set of trigger images is constructed based on the user specific key. The composition of the key is expressed as $K = k_1, k_2, ..., k_{n-1}$. Since the key is selected based on the user, the key itself is confidential. We utilize the key to construct a set of trigger images. A set of trigger images constitutes a set of trigger images to be used as the next stage of training. The set of trigger images is constructed based on the key, so the relationship between trigger images and labels is as follows:

$$\{Image, Label\} \leftarrow \{(I_i, k_i) \mid i = 0,1,2,...,n-1\}. \tag{1}$$

Where *Image* denotes the trigger image, *Label* denotes the label corresponding to the trigger image, and k_1 denotes the key. Here, in order to facilitate more convenient confirmation in the future, we directly take the key as the label corresponding to the trigger image. However, since the model itself has different classification tasks, the corresponding number of classifications is also different, we will put a limit on the key. The following is the key restriction for the n-categorization task model as an example.

$$ki \in [0,9],\ (i = 0,1,2,\ldots,n-1). \tag{2}$$

Where n represents the number of classifications of the model and also the number of trigger image sets and the length of the key.

Trigger Image Set Classification Training Phase. In this phase we will train the trigger image set using the original DNN model and the generated model, and fine-tune the original DNN model using the backdoor method. The main purpose of this phase is to enable the trigger image set to be accurately classified in the source DNN model with the same label provided by the key as shown in Equation 1. For the training method in this phase, we use the backdoor training method. The trigger image set and the clean training dataset are put into the source DNN model together for training. The specific method is as follows. First, we use the training set to train the original DNN model M_{origin} in the first round, so that the original DNN model reaches the required accuracy. Then, we mix the normal images in the training set and the trigger images in the trigger set to form a new training set. The new training set is input into M_{origin} for the second round of classification training.

The M_{origin} that has been trained using the training set has a high accuracy. So, when a new training set is input to the original DNN model is able to classify that dataset according to the labels. During the second round of training, M_{origin} performs normal classification on normal images. The newly added triggered images are misclassified. Here the loss due to classification error is called classification loss $\mathcal{L}_c$, for which the details are described in Section 3.3. The generative model uses the classification loss to fine-tune the trigger image, and then inputs the fine-tuned trigger image together with the normal image into the original DNN for classification training. After several repetitions of training the model to recognize the images in the training set and trigger set, the model gradually improves the classification accuracy of the trigger images under the training of the mixed dataset. The purpose of the second round of training is to enable the trigger images in the trigger image set to be accurately classified in the original DNN model. Mixing the normal images in the training set and the trigger images in the

trigger set for training can be used to assist the DNN model to classify the trigger image samples into the specified classification labels by using the correct image classification. It greatly improves the classification accuracy of the DNN model for the trigger image set. The model after this stage of training is denoted as M_{tra} .

Model Watermark Embedding Phase. We would like to mention a concept first. For a multi-classification task DNN model, the probability P of each classification can be obtained by Softmax function after classification, i.e., $P = \{P_j \mid j = 0, 1, ..., n-1(\sum P_j = 1)\}$, where P_j is denoted as the predicted probability of categorizing each category of the model in (0,1). For the prediction probability, if the variance of the prediction probability is larger then it represents the lower accuracy of the model in classifying the image. At this stage we utilize the fragile watermarking method to test fine-tuning only for the trigger set and embed the watermark into the model. The specific method is as follows; we merge the trigger graphs trained after the trigger image set classification training stage into the trigger set D_{tri}. We input D_{tri} into the DNN model M_{tra}, and perform Softmax operation to find the predicted probability of classification P_t for all the output classification results. Here, we only take the highest probability in the classification to calculate. After the predicted probability is calculated for all trigger images, the variance of the predicted probability $\sigma(P_t)$ is calculated. There are still misclassifications at this stage, so the classification loss $\mathcal{L}_c$ remains. This is where we arrive at the test loss $\mathcal{L}_t$ for this stage, which is described in more detail in Section 3.3. The generative model uses the test loss to fine-tune the trigger images in the trigger set, and then inputs the fine-tuned trigger images into M_{tra} for classification training. It is also necessary to input the test set into M_{tra} during the training process and record the classification accuracy of the test set. The training process will be repeated several times until specific conditions are met before stopping.

The main purpose of this phase is to make the trigger set classified in the DNN model accurate enough. That is, when the DNN model embedded with the watermark is modified, the model's classification of the images in the trigger set will produce errors. So the conditions for the end of training in this phase need to fulfill two requirements. First, sufficiently accurate classification in the DNN model means that the classification accuracy of the triggered images in the trigger set should be equal to 1, i.e., condition one is $Acc_{trigger} = 1$. Second, while we ensure that the DNN model can accurately recognize the trigger set, we also need to ensure the model's classification accuracy on the dataset that has normal images. So the second condition is to ensure that the classification accuracy of normal images in the test set is greater than or equal to the training accuracy of the DNN model on the training set, i.e., condition two is $Acc_{test} \geq Acc_{train}$. After the training process satisfies the above two conditions, the process stops and the watermark proposed in this paper is embedded into the DNN model by this training method. The model after embedding the watermark through this stage is denoted as M_{wm}

After the above three stages model M_{origin} is embedded with the watermark proposed in this paper. Finally, we will get the model M_{wm} embedded with the watermark and the trigger images to form the trigger set D_{tri}. Subsequently, the integrity and attribution of the model can be judged by M_{wm} and D_{tri}, and the details of the verification are in Section 3.4.

3.3 Watermark Training Loss

In this section, we focus on the losses mentioned in Section 3.2. The losses mentioned in Section 3.2 consist of two main ones. The first one is the classification loss $\mathcal{L}_c$ in the training stage of trigger set image classification. The second is the testing loss $\mathcal{L}_t$ in the model watermark embedding stage.

Classification Loss. This loss is generated during the training phase of the trigger set image classification. Since the trigger images and corresponding labels in the trigger set are not trained in M_{origin} . So when the trigger images and normal images are input into M_{origin} together for training, M_{origin} generates an error in classifying the trigger images. The error generated by M_{origin}'s inability to classify the trigger images to their corresponding labels, which we refer to as the classification error $\mathcal{L}_c$ is mentioned in Section 3.2. The multi-classification task DNN model can obtain the probability of each classification P_j by Softmax function after classification. the classification error is calculated based on the cross-entropy loss of the probability, i.e., $\mathcal{L}_c$ is the cross-entropy loss of multi-class classification, defined as follows:

$$\mathcal{L}_c = -\sum_{j=0}^{n-1} k_j \log(P_j). \tag{3}$$

Where n is the number of classifications of the multiclassification model, and P_j is denoted as the predicted probability of classifying each category of the model in (0, 1). The $\mathcal{L}_c$ exists both in the training phase of trigger set image classification and in the embedding phase of model watermarking.

Test Loss. This loss is generated in the model watermark embedding phase. This loss consists of two parts. The first one is the loss generated by M_{tra}'s classification of the triggered images in the trigger set, which is designed to correct the correctness of the classification of the triggered images. The second loss is the loss generated by the variance of the prediction probability, which is to adjust the model to be more accurate in predicting the probability of each classification of the triggered images. Here we reduce the variance of the classification prediction probability by calculation. Reducing the variance makes the classification prediction probabilities less discrete. It allows the watermarking model to make an error in the classification of the trigger image when it encounters a modification. The formula for calculating the variance of the prediction probability is given below:

$$\sigma(P_t) = \sum (k_j - \mu)^2 P_j. \tag{4}$$

$$\mu = \sum k_j P_j. \tag{5}$$

where $\sigma(P_t)$ is the variance of the predicted probability. μ denotes the mean of the i^{th}. classification. $\sum$ denotes the summation. P_j denotes the predicted probability of the j^{th} classification. After calculating the variance of the predicted probability, we can derive the loss formula in the embedding stage of the model watermark as:

$$\mathcal{L}_t = W\sigma(P_t) + \mathcal{L}_c, \tag{6}$$

where W is a weighting factor. This stage contains both parts of the loss. We add weighting coefficients to the variance loss part in order to improve the model's probability of classifying the triggered image while preventing overfitting.

The losses mentioned in this section are mainly used to improve the model's classification accuracy for triggered images in the trigger set. The purpose of using loss optimization is to achieve two things. The first is that the model has a classification accuracy of 1 for the triggered images, and the second is that the classification accuracy of the normal samples can reach the accuracy of the source model in the model. During the training process, to prevent overfitting caused by training, we consider variance in the loss. Overfitting can be better prevented by increasing the weight coefficients and adding variance to the overall loss.

3.4 Watermark Verification

The Authentication section in Fig. 1 shows the process of watermarking for authentication. In this section we give a detailed description of the process of watermark authentication.

Before presenting our watermarking authentication method, we describe the application scenarios of the watermarking method proposed in this paper. Consider the third aspect: model owner, user and attacker. The model owner has the highest administrative authority over the model and is also responsible for the security of the model. In order to reduce the leakage of the model during distribution or authorization. Model owners store their trained models in the Model Management Cloud Platform (MMCP), a model management platform that stores models and provides API interfaces. The platform enables models to be called by users without being distributed. This prevents models from being stolen during the distribution process. We will not go into details about MMCP here.

The API interface of MMCP provides a more convenient way to access the data, but the convenience of accessing the data may be utilized by attackers for poisoning. This results in model misclassification or model backdoor. Based on the above scenarios we can find. Attacks against the model will mainly be realized by uploading pictures through the API interface, i.e., poisoned pictures will be used by the attacker to realize poisoning attacks. The owner of the model integrity authentication also needs to be realized through the API interface.

In Section 3.2 we mentioned. After training the embedded watermark, we will get the embedded watermarked model $\mathscr{M}_{wm}$ and the trigger images to form the trigger set D_{tri}. Corresponding to the above scenario, $\mathscr{M}_{wm}$ is stored in MMCP by the owner, and D_{tri} is stored privately by the owner. In our approach, a new authentication metric

is proposed to verify whether the watermarked model has been modified or not. The metric is proposed based on the termination condition of the model watermark embedding phase.

The first metric is the classification accuracy $Acc_{trigger}$ of the trigger image. This metric is proposed based on one of the termination conditions in the watermark embedding phase of the model, $Acc_{trigger} = 1$. D_{tri} is input to the classification model through the API, and the classification accuracy of the triggered image is computed based on the returned triggered image classification result. If $Acc_{trigger} < 1$, then $\mathscr{M}_{wm}$ has been modified. The stronger the malicious attack against the model on $\mathscr{M}_{wm}$, the more $Acc_{trigger}$ drops.

The second metric is the classification label of the trigger set. After the watermarking training process in Section 3.2, $\mathscr{M}_{wm}$ is able to accurately classify the trigger images in D_{tri}. The trigger images and classification labels in D_{tri} correspond as shown in Eq. (1). The trigger images are classified a new kind of category in $\mathscr{M}_{wm}$ after $\mathscr{M}_{wm}$ suffers from a malicious attacker:

$$\mathcal{C} = c_i \mid \{i = 0,1,\ldots,n-1\}. \tag{7}$$

Where c_i denotes the category corresponding to the i^{th} graph after classification. The labeled category corresponding to the trigger image in Eq. (1) is k_i. Based on these two labeled values, we can determine whether the model has been tampered with by analyzing the corresponding values between the two. When there is a difference between c_i and k_i comparison, the more difference exists means the more $\mathscr{M}_{wm}$ has been modified.

4 Experiments

In the experiments of this paper, we focus on two datasets CIFAR-10 and CIFAR-100. Where CIFAR-10 is mainly trained on VGG-16, VGG-19 and ResNet-56 for classification tasks. CIFAR-100 is trained on ResNet-56 and DesNet-40 for classification tasks. In this paper, we describe the experimental setup in Sections 4.1, 4.2 and 4.3, respectively. Different experimental results and analysis are presented in Section 4.4.

4.1 Dataset and DNN

This section focuses on the datasets used in the experiments of this paper, which include CIFAR-10, CIFAR-100 and the trigger set proposed in this paper (see Table 1).

Table 1. Data set information used in this article

Datasets	Train	Test	Class	Resolution
CIFAR-10	50,000	10,000	10	32 × 32
CIFAR-100	50,000	10,000	100	32 × 32
TRIGGER SET	-	-	10	32 × 32

4.2 Implementation Details

The experiments are performed on Ubuntu 18.04 with an Intel Xeon Gold 5218 CPU @ 2.30GHz and an NVIDIA QUADRO RTX 8000 GPU. The watermark training scheme proposed in this paper is implemented in Python 3.9.2 and PyTorch 1.9.0 using trigger sets for model integrity detection. Next, we describe the training setup for the classification model.

Training Settings. VGG-16 and VGG-19 were trained on CIFAR-10 for 160 epochs, and ResNet-56 was trained on CIFAR-10 for 150 epochs. The optimizer used for these three models is SGD, with momentum set to 0.9 and the initial learning rate set to 0.1, reduced to 1e-4 at 100 epochs and 1e-5 at 140 epochs. ResNet-56 and DesNet-40 are trained on CIFAR-100 for 200 epochs. The initial learning rate is set to 1e-4 and reduced to 1e-5 at 160 epochs.

4.3 Evaluation Protocol

In terms of evaluation metrics, ω denotes the percentage of randomly selected normal images in the training set during the training phase of the trigger image set. *Baseline* is defined here as the classification accuracy of the model on the test set after training, and this metric is also used for the purpose of better watermarking impact on the accuracy of the model. Acc_{test} denotes the classification accuracy of the model on the test set after embedding the watermark. $Acc_{trigger}$ denotes the classification accuracy of the model on the trigger set after embedding the watermark. In this paper we will illustrate the choice about, ω by measuring *Baseline* and Acc_{test} as well as the time spent on training. Acc_{test} and $Acc_{trigger}$ are utilized to illustrate the effect of the model after being attacked by poisoning.

4.4 Results and Analysis

Selection of Training Set. The accuracies of the models trained on the test sets with different occupancy are demonstrated in Table 2. Two datasets, CIFAR-10 and CIFAR-100, were selected for this experiment and tested on VGG-16, VGG-19, and DesNet-40, respectively. Times represents the time spent on watermark embedding in the watermark embedding phase of the model, in which we set 10 epochs as one time.

In our experiments, we set the percentage of randomly selected normal images in the training set to *1%*, *5%*, *10%*, *12%* and *15%*, respectively. It is shown in the experimental results. As the percentage increases, the classification accuracy of the model on the test set after embedding the watermark is increasing and approaching *Baseline*. But the time required to embed the watermark is also increasing. So based on the analysis of the data in Table 2, we draw the following conclusions. In the training phase of triggering image set classification, the number of selected test sets is associated with the performance of the model and the efficiency of watermark embedding. When the number of selected datasets increases, the classification accuracy of the model for normal image test set keeps increasing to reach the level of *Baseline*. But the time required for watermark embedding also increases. Conversely, when the number decreases, the classification accuracy of the model on the test set decreases. But the time required for watermark embedding also decreases. In summary, we need to consider an appropriate number of selected training sets. Based on the data in Table 2, we will select *10%* of normal images from the training set with D_{tri}. Make sure to achieve a balance between model classification accuracy and watermark embedding time.

Table 2. Classification accuracy of test set in models trained on training sets with different occupancy ratios

Datasets	DNNs	Baseline (%)	ω (%)	Acc_{test} (%)	Times
CIFAR-10	VGG-16	93.74	1	80.62 (-13.12)	12
			5	84.34 (-9.40)	15
			10	**93.57 (-0.17)**	**16**
			12	93.70 (-0.04)	19
			15	93.69 (-0.05)	20
	VGG-19	93.62	1	82.66 (-10.96)	12
			5	83.63 (-9.99)	14
			10	**93.42 (-0.20)**	**16**
			12	93.56 (-0.06)	20
			15	93.56 (-0.06)	21
CIFAR-100	DesNet-40	74.86	1	63.62 (-11.24)	16
			5	68.83 (-6.03)	17
			10	**74.66 (-0.20)**	**20**
			12	74.66 (-0.20)	22
			15	74.79 (-0.07)	24

Model Watermarking Performance. In this section we will classify CIFAR-10 and CIFAR-100 in different models respectively. And compare the accuracy of DNN model after training with training set and the accuracy of DNN model after embedding watermarking. In order to illustrate the fidelity of the watermarking model proposed in this paper.

The output tensor of the final convolutional layer of the model is projected onto a two-dimensional plane, as shown in Fig. 2, using the visualization tool UMAP. The left and right images respectively depict the projected features extracted from the source model trained on CIFAR-10 and the black-box watermark model. The ten larger colored circles represent the projected feature points of 5,000 normal samples, while the purple dots within the circles represent the projected feature points of 36 trigger images, all belonging to ten image classes. As shown in Fig. 2, once the model embeds a fragile watermark, the projected feature points of trigger samples deviate from the center to the classification boundaries. When the feature projection points of trigger samples are close to the classification boundary, they possess higher potential for attacks, as even minor perturbations may lead to misclassification by the model. This enhances the stability and reliability of the black-box watermark technique, as trigger samples can still activate the watermark during the inference phase and influence the model's behavior, even in cases where direct access to the model parameters is not feasible.

The accuracy comparison of CIFAR-10 and CIFAR-100 in DNN model is shown in Fig. 3, respectively. It can be seen that the Acc_{test} after embedding the watermark in (B) will be slightly reduced than *Baseline*. But this reduction scale is very small is within reasonable fluctuation and does not affect the accuracy of the model. The main reason for this situation is that image classification in CIFAR-10 is relatively simple. Normal and triggered images are easily disturbed when they are trained together, which makes the DNN model classify normal images with a small decrease in accuracy. In (A) we can see that Acc_{test} is higher than *Baseline* part on ResNet-56. This shows that under the watermarking of the model proposed in this paper. There are cases where the accuracy of the watermarking model is slightly reduced and slightly improved. But both the decrease and increase are less than 0.2%.

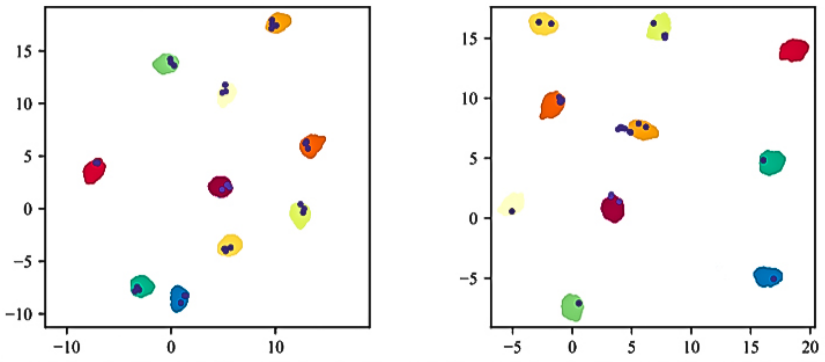

Fig. 2. The comparison of feature projections before and after watermark embedding

(The left and right images respectively depict the results obtained from training the source model and the black-box watermark model on CIFAR-10 dataset.)

Attacks Detection. In this section, we study the performance of watermarking techniques when attacked by contaminated datasets. We study the following two types of data poisoning behaviors. One type is traditional data poisoning. That is, the attacker intentionally introduces a large number of mislabeled samples in the training set to degrade the performance of the model. The other is backdoor data poisoning. The attacker uses mislabeled samples that contain the training set and with backdoor patterns as an attack. We prepared three types of datasets. The original dataset *origin*, the data poisoning dataset *poison* containing a dozen of mislabeled samples, and the backdoor data poisoning dataset *backdoor*. Use these three types of datasets to classify the data using $\mathscr{M}_{wm}$ and record the $Acc_{trigger}$ at different epochs.

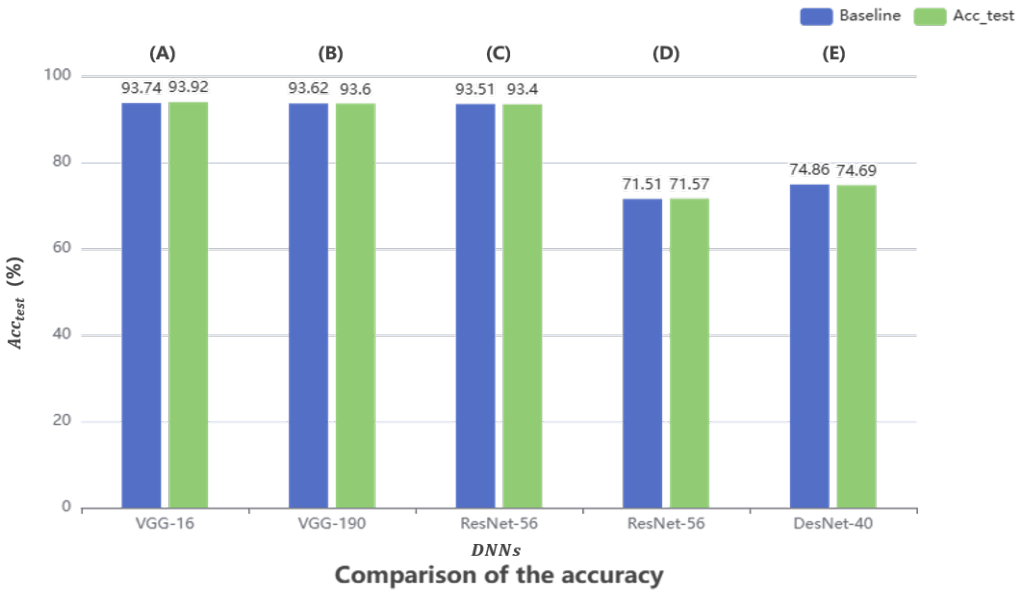

Fig. 3. Accuracy comparison

(Comparison of the accuracy of different DNN models in classifying normal images before and after embedding the watermark. x-axis is different DNN models and y-axis is Acc_{test} . The blue color represents the accuracy of the DNN model classification test set when the watermark is not embedded, and the green color is the accuracy of the DNN model classification test set after the watermark is embedded. Where (A), (B) and (C) are experimental data on CIFAR-10. (D) and (E) are experimental data on CIFAR-100.)

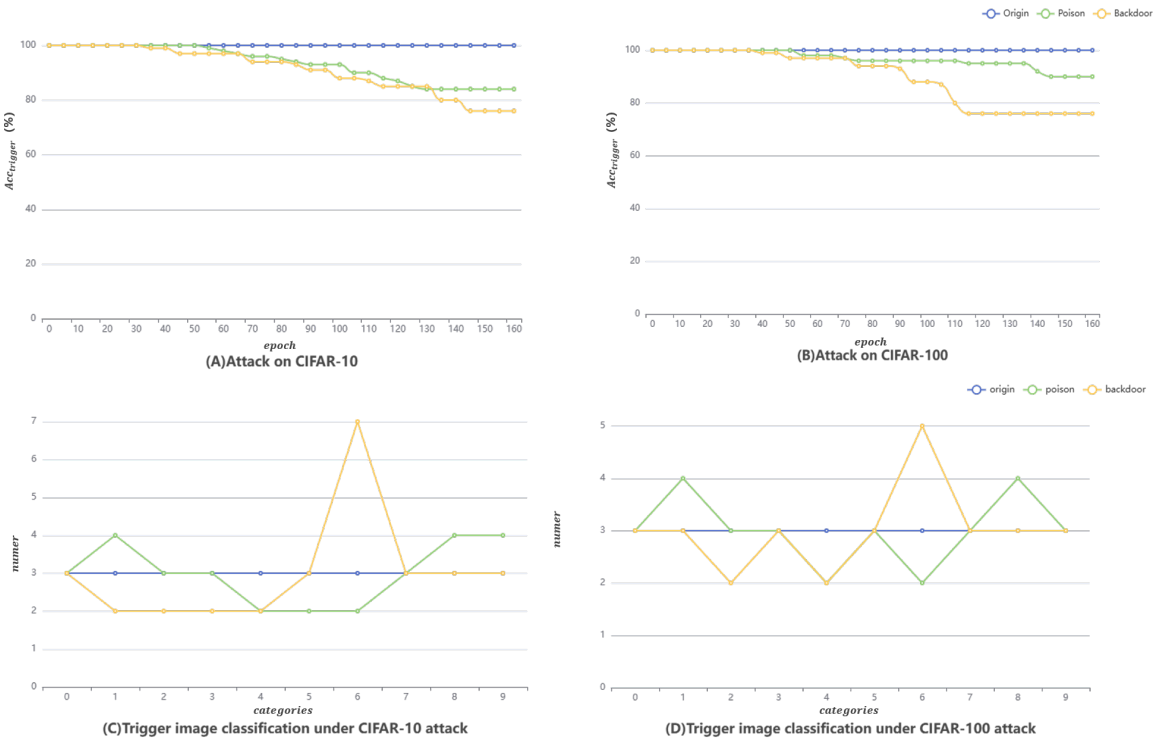

Fig. 4. Classification accuracy of triggered images after training with different attacks on CIFAR-10 and CIFAR-100 datasets

Fig. 4 show the experiments on the classification accuracy and classification categories of the trigger image, respectively. From Fig. 4 we can see. In *origin* is normal image and the model attacked with *origin* does not produce any modification to the model. Therefore, the accuracy in the experiment is consistent with $Acc_{trigger} = 1$. In Fig. 4 we see that the classification of the model attacked with *origin* also corresponds to the label corresponding to the trigger image. Thus, the experiment also conforms to our second judgment metric: label correspondence. Next, we analyze the remaining two attack scenarios with *origin* separately. About the attack on the *poison* dataset. First of all you can see in Fig. 4. After a period of attack, $Acc_{trigger}$ started to decrease continuously even down to almost 80%. And in Fig. 4 we can see that the categorization starts to change and different trigger images are misclassified into different categories. About *backdoor* dataset attack. First of all in Fig. 4 we can see. After a period of attack, $Acc_{trigger}$ produces a substantial decrease or even drops below 80%. In Fig. 4, it can be seen that a large number of trigger images are categorized into the same category.

The *poison* dataset attack uses multiple mislabeled samples, so it will present a situation where multiple trigger images are misclassified in Fig. 4. The *backdoor* dataset attack produces a situation where multiple trigger images are misclassified into the same class because it uses samples with backdoor patterns. In this experiment, the attack dataset we use has a relatively small amount of data, so the impact on the accuracy of $Acc_{trigger}$ is mild. And the reason we adopt a smaller amount of data is to verify whether the model watermark can be detected in the face of minor modifications. Table 3 also lists some data to determine if the model has been modified. In summary, the model watermarking proposed in this paper is capable of detecting model modifications generated by data poisoning and backdoor attacks. The usability of the model watermark in this paper is demonstrated.

Table 3. Trigger set precision and classification accuracy

(Class represents whether there is an error in the classification of the trigger set. √ indicates no classification error. × indicates a classification error. Indicators that satisfy the criteria for being recognized as a modified model are marked in the table.)

Datasets	DNNs	Acc_{test} (%)	Class	$Acc_{trigger}$ (%)	Status
CIFAR-10	VGG-16	93.74	√	**90.00**	Modified
		93.68	×	**86.67**	Modified
		93.75	×	100	Modified
		93.64	√	100	Safe
		84.88	×	**77.78**	Modified
CIFAR-100	DesNet-40	74.86	√	**93.50**	Modified
		74.80	×	**86.30**	Modified
		74.79	×	100	Modified
		74.83	√	100	Safe
		66.33	×	**73.00**	Modified

Watermark Frame Comparison. This section aims to compare and analyze the current mainstream centralized model protection watermarking frameworks. The selected mainstream watermark protection frameworks for comparison include backdoor watermarking, embedded passports, and adversarial samples. The main aspects of comparison include the impact of the model framework on model performance and whether the model can detect integrity. According to the data analysis in Table 4, it can be observed that white-box watermarking exhibits stronger capabilities in detecting model integrity. However, due to the adjustment of the model's structure, embedding watermarks may potentially affect the model's accuracy. Conversely, black-box watermarking has minimal impact on the model's accuracy. However, due to the embedding of watermarks based on backdoor training, detecting modifications to the model can be challenging.

Both the Embedding Passports and Adversarial Examples methods fall under the category of white-box watermarking, which affects the original deep neural network model, leading to a decrease in accuracy. On the other hand, although the Backdooring method does not impact the model's accuracy, it fails to detect the model's integrity. The method proposed in this paper has been experimentally tested and successfully detects model modifications while maintaining model accuracy.

Table 4. Comparison between different watermarking frameworks and the watermarking framework in this paper

(The test data set and model are CIFAR-10 and ResNet56.)

DNN	Watermark frame	Method type	Acc_{test} (%)	*Check*
ResNet-56	Baseline	-	93.51	-
	Embedding passports [25]	White box	↓	√
	Adversarial Examples [16]	White box	↓	√
	Backdooring [22]	Black box	=	×
	Ours	Black box	=	√

5 Conclusions

In this paper, we propose a neural network black-box watermarking method based on trigger set label loss, which effectively detects DNN model integrity. Experimental results show that the method is able to present the model when it is modified by the classification distribution of the trigger set, and does not impair the performance of the neural network on the original task. Our method is a novel detection method that can provide a good solution to the current commercial DNN modeling scenario. Since the method proposed in this paper is mainly implemented using the trigger image set and takes into account the classification accuracy for normal images. So, the watermarking method will have no effect on the classification accuracy of the model itself. When the model is modified, the classification of the triggered images will be affected thus producing wrong classification. Thus, we propose a judgment indicator for whether the model has been modified or not.

For the future outlook, we will continue to optimize for the model watermarking training phase. Try to reduce the time consumed for model embedding training. Optimize the trigger image to reduce the impact on classification.

In this paper, we introduce a novel neural network black-box watermarking method based on trigger set label loss, which effectively detects integrity breaches in DNN models. Experimental results demonstrate that our method accurately identifies model modifications through changes in the classification distribution of the trigger set, while preserving the neural network's performance on the original task. Our approach offers a promising solution for addressing integrity concerns in commercial DNN modeling scenarios. Since our proposed method primarily utilizes the trigger image set and considers classification accuracy for normal images, it does not affect the model's classification accuracy itself. When the model undergoes modifications, the classification of triggered images is affected, leading to incorrect classifications. Therefore, we propose a judgment indicator to determine whether the model has been modified.

In the future, our research will focus on optimizing the training phase of model watermarking to reduce the time required for embedding training. This optimization aims to streamline the process and make it more efficient. Additionally, we will concentrate on enhancing the trigger image to minimize its impact on classification accuracy. This involves refining the trigger image generation process to ensure that it does not introduce any unintended biases or distortions that could affect the model's performance. By refining both the training phase and the trigger image, we aim to further improve the overall effectiveness and robustness of our watermarking method.

6 Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 62166008), the Excellent Youth Science and Technology Talent in Guizhou Province(QKHPTRC-YQK[2023]028), the Central Government Guides Local Science and Technology Development Special Project (No. QKZYD[2022]4054), the Guiyang Science and Technology Talent Training Project (No. ZKHT[2023]48-9), the Academic Budding Talent Fund Project of Guizhou Normal University (No. QSXM[2022]31), and the Natural Science Research Project of Guizhou Provincial Department of Education (No. QJJ[2023]011).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Communications of the ACM 60(6)(2017) 84–90.

[2] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu, Convolutional neural networks for speech recognition, IEEE/ACM Transactions on audio, speech, and language processing 22(10)(2014) 1533–1545.

[3] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, G. Penn, Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition, in: Proc. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012.

[4] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, <https://arxiv.org/abs/1810.04805>, 2018 (accessed 21.10. 2021).

[5] J. Guo, M. Potkonjak, Watermarking deep neural networks for embedded systems, in: Proc. 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2018.

[6] M. Shafieinejad, N. Lukas, J. Wang, X. Li, F. Kerschbaum, On the robustness of backdoor-based watermarking in deep neural networks, in: Proc. Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security, 2021.

[7] M. Xue, Y. Zhang, J. Wang, W. Liu, Intellectual property protection for deep learning models: Taxonomy, methods, attacks, and evaluations IEEE Transactions on Artificial Intelligence 3(6)(2021) 908–923.

[8] C. Song, T. Ristenpart, V. Shmatikov, Machine learning models that remember too much, in: Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017.

[9] Y. Uchida, Y. Nagai, S. Sakazawa, S. Satoh, Embedding watermarks into deep neural networks, in: Proc. 2017 ACM on International Conference on Multimedia Retrieval, 2017.

[10] M. Kuribayashi, T. Tanaka, N. Funabiki, Deepwatermark: Embedding watermark into dnn model, J. 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, 2020.

[11] B. Cortiñas-Lorenzo, F. Pérez-González, Adam and the Ants: On the Influence of the Optimization Algorithm on the Detectability of DNN Watermarks, Entropy 22(12)(2020) 1379.

[12] M. Xue, S. Sun, C. He, ActiveGuard: An active intellectual property protection technique for deep neural networks by leveraging adversarial examples as users' fingerprints, IET Computers & Digital Techniques 17(3-4)(2023) 111-126.

[13] L. Fan, K. Ng, C.S. Chan, Digital passport: A novel technological strategy for intellectual property protection of convolutional neural networks, <https://arxiv.org/abs/1905.04368>, 2019 (accessed 11.04.2022).

[14] J. Tian, J. Zhou, J. Duan, Probabilistic selective encryption of convolutional neural networks for hierarchical services, in: Proc. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[15] X. Zhao, Y. Yao, H. Wu, X. Zhang, Structural watermarking to deep neural networks via network channel pruning, in: Proc. 2021 IEEE International Workshop on Information Forensics and Security (WIFS), 2021.

[16] X. Chen, C. Liu, B. Li, K. Lu, D. Song, Targeted backdoor attacks on deep learning systems using data poisoning, <https://arxiv.org/abs/1712.05526>, 2017 (accessed 08.11.2021).

[17] D. Hitaj, B. Hitaj, L.V Mancini, Evasion attacks against watermarking techniques found in mlaas systems, in: Proc. 2019 Sixth International Conference on Software Defined Systems (SDS), 2019.

[18] Y. Adi, C. Baum, M. Cisse, B. Pinkas, J. Keshet, Turning your weakness into a strength: Watermarking deep neural networks by backdooring, in: Proc. 2018 USENIX Security Symposium (USENIX Security 18), 2018.

[19] Z. Li, C. Hu, Y. Zhang, S. Guo, How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn, in: Proc. 2019 Annual Computer Security Applications Conference, 2019.

[20] H. Wu, G. Liu, Y. Yao, X. Zhang, Watermarking neural networks with watermarked images, IEEE Transactions on Circuits and Systems for Video Technology 31(7)(2020) 2591–2601.

[21] L. Wang, Y. Song, D. Xia, Deep neural network watermarking based on a reversible image hiding network, Pattern Analysis and Applications 26(3)(2023) 1–14.

[22] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: Proc. 2017 IEEE Symposium on Security and Privacy, 2017.

[23] Y. Xiang, L. Gao, J. Zhang, L. Zhang, Q. Zhong, Protecting ip of deep neural networks with watermarking: A new label helps, in: Proc. 2020 Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2020

[24] D. Xia, L. Wang, Y. Song, X. Lou, Review of deep neural network model digital watermarking technology, Science Technology and Engineering 23(5)(2023) 1799–1811.

[25] L. Fan, K.W. Ng, C.S. Chan, Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks, Advances in neural information processing systems 32 (2019) 4714–4723.