ActiveGuard: Active Intellectual Property Protection for Deep Neural Networks via Adversarial Examples based User Fingerprinting

Mingfu Xue[1]*, Shichang Sun[1], Can He[1],
Dujuan Gu[2], Yushu Zhang[1], Jian Wang[1], Weiqiang Liu[3]
[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China.
[2]NSFOCUS Information technology CO., LTD, China
[3]College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, China.
{mingfu.xue, sunshichang, hecan, yushu, wangjian, liuweiqiang}@nuaa.edu.cn, gudujuan@sina.com

Abstract

The training of Deep Neural Networks (DNN) is costly, thus DNN can be considered as the intellectual properties (IP) of the model owners. To date, most of the existing works protect the copyright of DNN through watermarking. However, the DNN watermarking is a passive verification method that can only work after the DNN model is pirated. In this paper, we propose an active DNN copyright protection method against DNN piracy, named *ActiveGuard*. *ActiveGuard* aims to achieve active authorization control and users' identities management for DNN, and can provide ownership verification. Specifically, *ActiveGuard* exploits the elaborate adversarial examples as users' fingerprints to distinguish authorized users from unauthorized users. Legitimate users can enter fingerprints into DNN for identity authentication and authorized usage, while unauthorized users will obtain a poor model performance. In addition, *ActiveGuard* enables the model owner to embed a watermark into the weights of DNN for ownership verification. Compared to the few existing active protection works, *ActiveGuard* is the first work to achieve both active authorization control and users' identities identification. Besides, *ActiveGuard* induces lower overhead than these existing works. Experimental results on LeNet-5 and Wide Residual Network (WRN) models demonstrate the effectiveness and robustness of the proposed method.

1 Introduction

Training a high-performance Deep Neural Networks (DNN) is costly and time-consuming (Uchida et al. 2017; Rouhani, Chen, and Koushanfar 2019; Chen and Wu 2018; Xue, Wang, and Liu 2021). Therefore, the trained DNN model can be regarded as a valuable intellectual property (IP) of the model owner. However, malicious users may illegally copy, redistribute, or abuse the models without permission (Chen and Wu 2018; Lin et al. 2021; Xue, Wang, and Liu 2021; Zhang et al. 2018). The IP protection for DNN is an emerging problem, which has attracted more and more serious concerns. Existing copyright protection methods in the multimedia field cannot be applied to DNN copyright protection directly (Xue, Wang, and Liu 2021; Chen et al. 2019).

In recent years, many works have been proposed to protect the copyright of DNN using DNN watermarks. However, the DNN watermarking is a passive verification method that can only work after the model is pirated, which cannot prevent piracy in advance. Besides, the DNN watermarking method is unable to identify/manage different users' identities, which cannot meet the requirements of practical commercial applications.

To date, few active authorization control works (Chen and Wu 2018; Fan, Ng, and Chan 2019; Chakraborty, Mondal, and Srivastava 2020) have been proposed to protect the copyright of DNN models, in which the authorized users can obtain a high accuracy when using DNN, while illegal users will obtain a poor accuracy. However, the work (Chen and Wu 2018) requires an extra anti-piracy transformation module to verify whether a user is legal or not. To use the DNN normally, an authorized user requires to preprocess each input data using the transformation module, which introduces high computational overhead. The work (Fan, Ng, and Chan 2019) embeds multiple passport layers into the DNN, which will introduce high overhead. Besides, the passport-based method is vulnerable to tampering attack and reverse-engineering attack. The hardware-assisted method (Chakraborty, Mondal, and Srivastava 2020) relies on the trusted hardware devices (as a root-of-trust) to store the key for each user, which is costly for commercial applications. Further, all these existing active authorization control methods (Chen and Wu 2018; Fan, Ng, and Chan 2019; Chakraborty, Mondal, and Srivastava 2020) do not support users' fingerprints management, which makes them unsuitable for commercial applications.

In this paper, we aim at actively protecting the copyright of DNN models and providing users' identities management, which can prevent the occurrence of DNN piracy in advance and manage users' identities. We propose an active IP protection method for DNN via adversarial examples based user fingerprinting, named *ActiveGuard*. The proposed method is able to achieve active authorization control, users' fingerprints management, and ownership verification. To realize users' fingerprints management, *ActiveGuard* generates specific adversarial examples as users' fingerprints, and each authorized user is assigned with an adversarial example as his fingerprint. Then, authorized user can input his unique adversarial example into the DNN to

*Corresponding author: Mingfu Xue

verify his identity. In order to achieve authorization control, *ActiveGuard* adds a control layer to DNN. The control layer can constrain the usage of the unauthorized users on a protected DNN model, i.e., make the DNN dysfunctional to unauthorized users. In order to realize ownership verification, *ActiveGuard* embeds a numerical watermark into DNN's weights with a parameter regularizer.

The contributions of this paper are four-folds:

- For the first time, we propose an active IP protection method for DNN via adversarial examples based user fingerprinting. *ActiveGuard* regards adversarial examples with specific classes and confidences as users' fingerprints, and achieves authorization control based on the uniqueness of each user's fingerprint.
- The proposed *ActiveGuard* supports ownership verification for suspicious DNN models. We design an effective DNN watermarking scheme, where a numerical watermark can be embedded in DNN's weights discretely by leveraging a regularizer. This watermarking scheme can successfully embed a large-capacity watermark in DNN's weights without affecting the normal usage of the DNN model. Compared with the existing watermarking method (Uchida et al. 2017), the proposed watermark embedding method can provide a larger capacity (0$\sim$9 each bit, rather than 0$\sim$1 each bit) and is more stealthy (can be embedded discretely).
- Most of the existing works are passive verification methods, while this work can provide active copyright protection and copyright management for DNN. Compared with the few existing active authorization control works (Chen and Wu 2018; Fan, Ng, and Chan 2019; Chakraborty, Mondal, and Srivastava 2020), *ActiveGuard* is the first work to achieve both authorization control and users' identities management. Besides, *ActiveGuard* induces lower overhead than these existing works.
- *ActiveGuard* is demonstrated to be robust to model fine-tuning attacks and model pruning attacks.

2 The Proposed Method

2.1 Overall Flow

As shown in Figure 1, the overall flow of the proposed method can be divided into the following steps: (i) The model owner embeds the specific numerical watermark into the DNN model. (ii) The model owner deploys the watermarked DNN as an online service, and generates the licenses (i.e., the adversarial examples) for authorized users to achieve active authorization control. (iii) The authorized users submit fingerprints (adversarial examples) to DNN model to verify their identities, and then use the DNN normally. On the contrary, unauthorized users will obtain a low performance due to the added control layer. (iv) When the model owner suspects that the DNN has been pirated, he can extract the embedded watermark from the weights of specific convolutional layer of the suspicious DNN model. If the watermark can be successfully extracted, the ownership of the suspicious DNN can be verified.

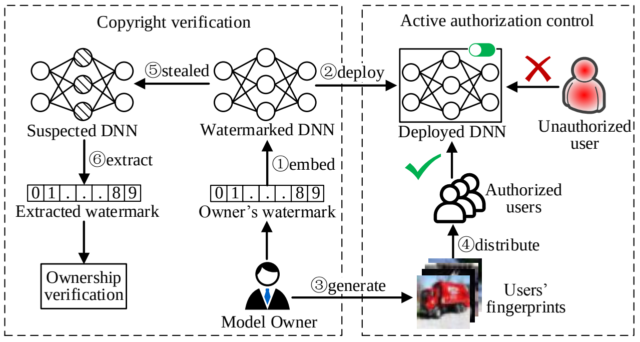

Figure 1: Overview of the proposed active intellectual property protection method for DNN.

The proposed *ActiveGuard* method has four functions (authorization control, users' fingerprints generation, users' fingerprints management, copyright verification), which will be described in Section 2.2$\sim$2.5, respectively:

2.2 Authorization Control

The procedure of active authorization control can be divided into three steps.

1) An adversarial example is assigned to an authorized user as his fingerprint. The proposed *ActiveGuard* method generates each adversarial example based on a specific class t with a fixed confidence c. In this way, each adversarial example is unique, thus can represent the unique identity of each authorized user.
2) Users submit their fingerprints to DNN for identity authentication before using the DNN model. Specifically, we design a control layer and add it to the end of the DNN to restrict the usage of unauthorized users on the DNN.
3) For authorized users with legal fingerprints, the *ActiveGuard* will reload the DNN model, and the added control layer will be automatically removed. As a result, the authorized users can use the DNN model normally (without the control layer). However, for unauthorized users, the DNN with the control layer will output randomly predicted results.

The proposed *ActiveGuard* exploits the difference of confidences to distinguish authorized users from unauthorized users. In general, for a DNN with high performance, the clean inputs will be classified as ground-truth labels with high confidences, while the well-crafted adversarial examples will be classified as their target classes with high confidences. In other words, very few inputs will be classified as a class t with a low confidence (below 0.50). Inspired by the above observation, this paper utilizes the low confidence interval (ranges from 0.10 to 0.50) to achieve active authorization control. First, we select some fixed confidences (such as 0.20, 0.30 and 0.40) from the low confidence interval $[0.10, 0.50)$, and use these selected confidences to construct a set C_{fp}, i.e., $C_{fp} = \{0.20, 0.30, 0.40\}$. Second, we generate such adversarial examples that are classified by the DNN as the specific target classes, while their classification confidences are in the set C_{fp}. In this way, when a user submits his fingerprint (i.e., adversarial example) to the protected DNN, he would be regarded as an authorized user if his fingerprint is classified as a specific target class with a confidence in the set C_{fp}. Otherwise, the user is considered to be unauthorized.

In addition, to achieve access control, we design a control layer based on the *Lambda Layer* (Chollet et al. 2015), and add the control layer to the end of the DNN. The control layer involves several control conditions and tensor (multi-dimensional vector) operations, and the implementation of the layer is as follows: (i) Receive the predicted class and the confidence vector propagated by the output layer, and extract the highest confidence in the confidence vector. (ii) Calculate the errors between the highest confidence and each confidence in the set C_{fp}. The minimal error among these calculated results is denoted as E_c. If E_c is less than the tolerable error, the user is considered to be an authorized user. Otherwise, the user is considered to be an unauthorized user. (iii) Output a randomly predicted result to the unauthorized user. For authorized users, the model will be reloaded automatically and the added control layer will be removed to provide normal performance.

2.3 Users' Fingerprints Generation

We introduce how to generate users' fingerprints (i.e., specific adversarial examples). Formally, the protected DNN has K classes and T assignable confidences (i.e., the set C_{fp} has T elements). In this way, a total of $K \times T$ fingerprints can be assigned to users. The fingerprint of a user is denoted as f, and all the $K \times T$ fingerprints constitute the fingerprint library FP, i.e., $FP = \{f_1, f_2, \ldots, f_{K\times T}\}$. The set of K classes is represented by $L_{fp} = \{0, 1, 2, \ldots, K-1\}$, and the set of T confidences is denoted as $C_{fp} = \{c_1, c_2, \ldots, c_T\}$.

In this paper, to ensure that a generated adversarial example is unique to represent the fingerprint of each authorized user, the method to generate the adversarial examples should satisfy the following goal: the generated adversarial example should be classified as the target class t with an fixed confidence c when it is input into the model M, where t is a class randomly selected from the set L_{fp} and c is a confidence in C_{fp}. To this end, this paper generates the adversarial examples using the ***C&W*** method (Carlini and Wagner 2017). The optimization function of ***C&W*** method is as follows (Carlini and Wagner 2017) :

$$\text{minimize } \{||\frac{1}{2}(\tanh(\delta)+1)-x||_2^2+\alpha\cdot g(\frac{1}{2}(\tanh(\delta)+1)\} \tag{1}$$

where x is the input, δ is a variable to craft the perturbation, and α is a constant used to control the magnitude of the perturbation. The first term $||\frac{1}{2}(\tanh(\delta)+1)-x||_2^2$ denotes the L_2 norm that measures the difference between a clean image and the generated adversarial example. The second term $g(\frac{1}{2}(\tanh(\delta)+1)$ is the objective function, where $g(\cdot)$ is originally defined as follows (Carlini and Wagner 2017):

$$g_0(x') = \max(\max\{Z(x')_k : k \neq t\} - Z(x')_t, 0) \tag{2}$$

where x' is an adversarial example, t is the target class, and Z is the unnormalized score from the penultimate layer of the DNN (Carlini and Wagner 2017). In this paper, to generate users' fingerprints, a confidence control term $||Z(x')_t - c||$ is added to the objective function, where c is a fixed confidence. Therefore, the objective function $g(\cdot)$ in this paper can be formalized as follows:

$$g(x') = g_0(x') + ||Z(x')_t - c|| \tag{3}$$

The $g_0(x')$ ensures that the generated adversarial example x' is classified as the class t. The confidence control term $||Z(x')_t - c||$ guarantees that the confidence on target class t is close to the predefined value c, where c is a legal confidence in the set C_{fp}.

The proposed *ActiveGuard* generates effective adversarial examples by solving the above Equation (1) with the $g(\cdot)$ defined in Equation (3).

2.4 Users' Fingerprints Management

Users' fingerprints allocation. In *ActiveGuard*, a unique fingerprint is assigned to each user based on the class and the confidence. In other words, the fingerprint of a user will be uniquely determined by a class t with a confidence c. The legal combination of class t and confidence c is denoted as *Fingerprinting Output* (FO), i.e., FO $= \{(t, c)|t \in L_{fp}, c \in C_{fp}\}$. Therefore, the users' fingerprints allocation can be implemented by assigning users with different FOs, as follows:

- The FO $= (i, c_j)$ is assigned to user $(i \cdot T + j)$, where $i \in \{0, 1, \ldots, K-1\}$, $j \in \{1, 2, \ldots, T\}$.

First, according to all K class labels and T predefined confidences, a total of $K \times T$ legal FOs are obtained. Second, the users' fingerprints are generated based on the above FOs. Finally, the generated fingerprints are allocated to authorized users, and each authorized user will acquire an fingerprint. For the protected DNN model, the authorized user with fingerprint $f_{i\cdot T+j}$ will be classified as class i with the confidence c_j.

Users' fingerprints authentication. Based on the prediction of the model, the proposed method can determine the legitimacy of each user and determine the identity of each authorized user. There will be a slight error between the output confidence $P_{max}(f)$ and c during the users' fingerprints authentication, i.e., $P_{max}(f) \approx c$, where $P_{max}(f)$ represents the maximal probability in the K-dimensional probability vector outputted by model f. We denote the tolerable error of the confidence as ε. If the confidence $P_{max}(f)$ output by DNN is within the error range $(c - \varepsilon, c + \varepsilon)$, the $P_{max}(f)$ would be considered to be matched with c, i.e., the user passes the identity authentication.

2.5 Copyright Verification

The adversarial example based method is difficult to distinguish the model owner from the authorized users. Therefore, we propose a watermarking method for copyright verification. Inspired by the watermark embedding method in work (Uchida et al. 2017), this paper embeds a n-digits watermark into the weights of DNN's convolutional layer for copyright verification. Compared to the watermarking method in (Uchida et al. 2017), our proposed watermark embedding method has two significant advantages: (i) First, the watermark in work (Uchida et al. 2017) consists of binary strings (0/1), where each bit of a watermark can only be embedded with two different digits (0 or 1). The proposed method extends the form of watermarking to numerical digits 0-9 through a linear mapping. This allows each bit of a watermark can be embedded with 10 different digits (i.e., 0-9), which greatly improves the capacity of watermark embedding. (ii) Second, the proposed method can embed the watermark discretely, i.e., the watermark can be embedded into discontinuous weights of a DNN model, while the watermark in work (Uchida et al. 2017) is embedded in consecutive positions. As a result, our embedded watermark is more stealthy (more difficult to be noticed) and more flexible.

The process of the proposed copyright verification method includes the following two parts: watermark embedding, watermark extraction and verification.

Watermark Embedding First, the proposed *ActiveGuard* embeds a n-digits watermark $\mathbf{wm} = (d_1, d_2, ..., d_n)$ into the weights of the target DNN model M. In this paper, the watermark is embedded into a specific convolutional layer of DNN. For a DNN, the weight matrix of a convolutional layer can be denoted as a 4-dimensional tensor $\mathbf{D} = (F, F, I, O)$, where F is the size of the convolution kernel, I is the number of input channels, and O is the number of output channels (Uchida et al. 2017; Chen et al. 2019). In this paper, the watermark is embedded into the maximum component among all O components of tensor $\mathbf{D}$, where each component is a tensor in the form of (F, F, I). In this way, a total of $F \times F \times I$ positions are available to embed the watermark, and the weights at these m ($m = F \times F \times I$) positions are denoted as a vector $\mathbf{w}$. We define another weight vector $\mathbf{v}$ to represent the weights at the n ($n < m$) randomly selected positions where the watermark is embedded.

Second, this paper aims to embed n-digits watermark $\mathbf{wm}$, i.e., the watermark consists of a series of numerical digits (0-9). Generally, in order not to affect the performance of the target DNN model, the weights of DNN after embedding the watermark should be close to the original weights. To this end, we design a watermark mapping function *map(·)*, which can map the weight vector $\mathbf{v}$ (small values) to the watermark vector $\mathbf{wm}$ (large digits). In other words, the designed watermark mapping function linearly amplifies the weight values (e.g., 0.22, 0.34) so as to map them to watermark digits (0-9). The *map(·)* is a linear function and can be formalized as $d = ah + b$, where h is a weight value, d is a digit value of the watermark, a and b are constants.

Finally, in order not to affect the performance of DNN models, the parameter regularizer (such as L_1-norm or L_2-norm) can be applied to embed the watermark (Uchida et al. 2017). In this paper, the proposed *ActiveGuard* exploits the mean square error (Allen 1971) to calculate the loss between the embedded watermark $\mathbf{wm}$ and the vector $map(\mathbf{v})$. The calculated result will be added to the original loss function of the DNN as a parameter regularizer, which aims to constrain the influence on performance that caused by the watermark embedding. In this way, the final loss function L of the target DNN model can be formalized as follows:

$$L = L_0 + \lambda \frac{1}{n} \sum_{k=1}^{n} (d_k - map(v_k))^2 \tag{4}$$

where L_0 is the original loss function. The $d_k \in \mathbf{wm}$ is the k-th digit of the watermark, and $v_k \in \mathbf{v}$ is the weight value in the position corresponding to the watermark digit d_k. Besides, $\frac{1}{n} \sum_{k=1}^{n} (d_k - map(v_k))^2$ is the mean square error (Allen 1971) between $map(\mathbf{v})$ and $\mathbf{wm}$, and λ is a parameter to adjust the term.

Watermark Extraction and Verification The proposed process of watermark extraction and verification are as follows. It takes the suspected model M', the target convolutional layer l, the watermark $\mathbf{wm}$, the positions $\mathbf{p}$ of the embedded watermark, and the watermark mapping function $map(\cdot)$ as inputs, and outputs the verified result R.

Step 1. The model owner obtains the parameters of the suspected model M', i.e., the target convolutional layer l, and the weights $\mathbf{D}_l$ of the layer l.

Step 2. The model owner utilizes the positions $\mathbf{p}$ of embedded watermark to extract the weights $\mathbf{v}$ of target convolutional layer l at these corresponding positions.

Step 3. The model owner exploits the function $map(\cdot)$ to map these extracted weights to the watermark digits, and compares the mapping result $\mathbf{wv_p}$ with his watermark $\mathbf{wm}$.

In Step 3, since the values of weights are floating-point numbers, we round each digit of the $\mathbf{wv_p}$ to an integer.

3 Experimental Results

3.1 Experimental Setup

In our experiments, we evaluate the proposed *ActiveGuard* on the MNIST (LeCun, Cortes, and Burges 1998) and CIFAR-10 (Krizhevsky 2009) datasets. We train the LeNet-5 (Lecun et al. 1998) model on the MNIST (LeCun, Cortes, and Burges 1998) dataset, and train the Wide Residual Network (WRN) (Zagoruyko and Komodakis 2016) model on the CIFAR-10 (Krizhevsky 2009) dataset.

Users' fingerprints settings. For users' fingerprints allocation or authentication, the number of authorized users that can be supported is calculated as follows. As mentioned in Section 2.4, a user's fingerprint is considered to be legal if the confidence of this fingerprint is between $c - \varepsilon$ and $c + \varepsilon$. In this way, the error interval of each authorized user is 2ε. Given K different classes, the tolerable error ε and the confidence interval $[z_1, z_2)$, the total number N_{au} of authorized users that the *ActiveGuard* method can support is calculated as follows:

$$N_{au} = K \times (z_2 - z_1)/2\varepsilon \tag{5}$$

In our experiments, $K = 10$ (10 classes), and the tolerable error ε of the confidence is set to be 0.01. Therefore, if the confidence interval for authorized users is

Dataset	Confidence c	Class label t									
		0	1	2	3	4	5	6	7	8	9
MNIST	0.20	99%	99%	99%	99%	96%	100%	99%	99%	100%	100%
	0.30	98%	100%	100%	100%	97%	100%	98%	98%	100%	99%
	0.40	99%	100%	100%	100%	97%	100%	98%	100%	100%	100%
CIFAR -10	0.20	100%	100%	100%	100%	100%	100%	100%	99%	100%	100%
	0.30	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	0.40	100%	99%	100%	100%	100%	100%	100%	99%	100%	100%

* Each combination of (t, c) represents the identity of an authorized user.

Table 1: Authentication Success Rates of 30 Different Combinations of Class Label t and Confidence c

Dataset	Model	Epoch	Test accuracy	Accuracy drop	V_{owner}
MNIST	LeNet-5 without watermarks	50	99.12%	N/A	N/A
	Watermarked LeNet-5 (by training from scratch)	50	99.15%	**-0.03%**	**success**
	Watermarked LeNet-5 (by fine-tuning)	20	99.12%	**0**	**success**
CIFAR-10	WRN without watermarks	200	91.38%	N/A	N/A
	Watermarked WRN (by training from scratch)	200	91.46%	**-0.08%**	**success**
	Watermarked WRN (by fine-tuning)	30	91.38%	**0**	**success**

Table 2: Accuracy and Ownership Verification Results on MNIST and CIFAR-10 Datasets

$[0.10, 0.50)$, the proposed *ActiveGuard* can support up to 200 $(10 \times (0.50 - 0.10)/0.02)$ authorized users. Specifically, under the above setting, the candidate confidence set is $\{0.10, 0.12, 0.14, 0.16, \ldots, 0.46, 0.48\}$, from which we can choose some confidences to construct the set C_{fp}. For the sake of simplicity, in the experiments, we only choose three confidences (i.e., 0.20, 0.30, 0.40) to construct the set C_{fp}. In this way, $C_{fp} = \{0.20, 0.30, 0.40\}$, $K = 10$, and $T = 3$. As a result, *ActiveGuard* assigns fingerprints for 30 authorized users in the experiments.

Watermarking settings. As discussed in Section 2.5, the watermark is embedded into a convolutional layer of DNN. As discussed in Section 2.5, the weight structure of the weight matrix $\mathbf{D}$ is (F, F, I, O), and the maximum watermark length is calculated by $F \times F \times I$. For example, we can embed at most 150 (i.e., $5 \times 5 \times 6$) digits into the weights at conv 2 layer ($D = (5, 5, 6, 16)$) of the LeNet-5 model (Lecun et al. 1998).

In our experiment, the length of the watermark is set to be 13. The watermark is embedded at the position $\mathbf{p}$ in the conv 2 layer, where $\mathbf{p}$ is randomly selected from all 150 (LeNet-5) and 576 (WRN) positions in the conv 2 layer. Additionally, the range of weight at the conv 2 layer of the LeNet-5 model is $[0.10, 0.45]$. Therefore, the watermark mapping function can be calculated, and the result is $d = (180/7)h - (18/7)$. Similarly, for the WRN model (Zagoruyko and Komodakis 2016), the range of weight at the conv 2 layer is $[0.20, 1.10]$, thus the watermark mapping function is $d = 10h - 2$. We follow the settings in work (Uchida et al. 2017) to set the constant λ to be 0.01.

3.2 Authorization Control Performance

Figure 2 shows the test accuracy for authorized usage and unauthorized usage on the two datasets. It is shown that, the test accuracy for authorized users on the two datasets is 99.15% (MNIST (LeCun, Cortes, and Burges 1998)) and 91.46% (CIFAR-10 (Krizhevsky 2009)), respectively. However, the performance of the unauthorized usage is much lower, as the test accuracy is only 8.92% (MNIST dataset) and 10% (CIFAR-10 dataset), respectively. Therefore, the proposed *ActiveGuard* method can achieve active authorization control, and can effectively prevent DNN models from being illegally used. The reason why the test accuracy for unauthorized users is close to 10% is as follows. The control layer of the DNN outputs a randomly chosen class to the unauthorized users. Hence, the test accuracy for the unauthorized users is equivalent to the accuracy of randomly guessing a class from all K classes ($K = 10$).

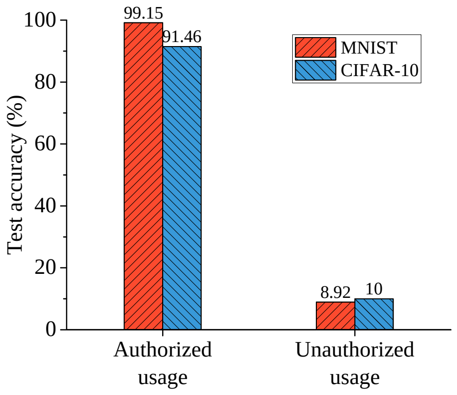

Figure 2: Test accuracy for authorized usage and unauthorized usage on the MNIST and CIFAR-10 datasets.

3.3 Users' Fingerprints Management Performance

As discussed in Section 3.1, in the experiments, there are 10 different classes ($L_{fp} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$) and 3 different confidences ($C_{fp} = \{0.20, 0.30, 0.40\}$). In this way, there are a total of 30 (10×3) FOs. Each FO is a combination of class label t and confidence c, and each FO corresponds to an authorized user. Note that, in our experiments, to evaluate the effectiveness of the proposed *ActiveGuard* method, we generate 100 different fingerprints for each authorized user to calculate the authentication success rate of an authorized user. However, when deploys the DNNs in real world, the model owner only requires to generate one adversarial example for each authorized user, and each user performs identity authentication by submitting one fingerprint.

Table 1 reports the fingerprint authentication success rate of 30 authorized users. For MNIST dataset (LeCun, Cortes, and Burges 1998), the fingerprint authentication success rate of 30 users is 96% at the lowest, 100% at the highest. For CIFAR-10 dataset (Krizhevsky 2009), the minimum and maximum fingerprint authentication success rates of 30 users is 99% and 100%, respectively, and 27 users achieve the success rates of 100%. It is shown that, all authorized users can successfully pass the authentication with a high success rate.

3.4 Copyright Verification Performance

In the experiment, we embed a 13-digits watermark $\mathbf{wm}_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 2, 1, 0]$ to protect the copyright of DNN model. For watermarks with other lengths, the proposed watermark embedding method is also feasible. We embed the watermark with two different approaches: 1) embedding the watermark by training from scratch; 2) embedding the watermark by fine-tuning. This paper defines a metric named V_{owner} to evaluate the performance of ownership verification. If the watermark is successfully extracted from the target layer of DNN model, $V_{owner} = success$, otherwise, $V_{owner} = failure$. The results of ownership verification are shown in Table 2. It is shown that, the watermark can be successfully extracted to verify the ownership of the model, regardless of the two embedding approaches (training from scratch or fine-tuning). Meanwhile, the normal performance of the watermarked DNN is similar to the performance of the DNN without watermarks. This indicates that, the proposed *ActiveGuard* method will not affect the normal performance of the DNN after embedding the watermark.

3.5 Robustness of the Proposed Method

We evaluate the robustness of the proposed method against fine-tuning (Simonyan and Zisserman 2015; Pittaras et al. 2017) attack and model pruning (Han et al. 2015) attack.

Table 3 shows the test accuracy and ownership verification of watermarked DNN under the fine-tuning attack. After 30 and 50 epochs of fine-tuning, the test accuracy of the LeNet-5 model is 99.53% and 99.53% respectively, while the test accuracy of the WRN model is 91.47% and 91.53% respectively. The test accuracy of the watermarked DNN before and after different epochs of fine-tuning attack is consistent. In other words, the proposed *ActiveGuard* method is robust against the fine-tuning attack.

For the model pruning attack, in our experiments, we assume that the adversary has known the layer where the watermark is embedded, which is a strong attack assumption. We adopt the pruning method in work (Han et al. 2015) to prune the target layer (i.e., the layer where the watermark is embedded) of the watermarked DNNs (LeNet-5 and WRN). For the target layer, the $r\%$ weights with the smallest absolute values are pruned, where $r\%$ is the pruning rate. The pruned weights are set to be 0. The watermark embedded into the LeNet-5 and WRN models is $\mathbf{wm}_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 2, 1, 0]$.

The robustness of the watermarked DNN against the pruning attack is shown in Table 4. First, as the pruning rate increases, the test accuracy of LeNet-5 and WRN models decreases sightly. However, even 60% of the weights in watermarked DNN are pruned, the test accuracy of two models still maintains at 98.54% (LeNet-5) and 89.87% (WRN), respectively. Besides, the embedded watermark performs well in terms of ownership verification. Specifically, even 90% weights are pruned, the watermark embedded into the WRN model can still be successfully extracted. However, the ownership verification fails when 50% weights in LeNet-5 model are pruned. The reason is that, the LeNet-5 (Lecun et al. 1998) model is smaller with fewer parameters. Therefore, when pruning, the smaller values in the watermark are more likely to be pruned. We also evaluate the robustness of the watermarked LeNet-5 mdoel against the pruning attack with a watermark with larger values. Specifically, the watermark $\mathbf{wm}_2 = [3, 8, 7, 6, 8, 7, 6, 9, 9, 4, 8, 6, 5]$ is embedded into the LeNet-5 model for evaluation. As shown in Table 4, for LeNet-5 model embedded with $\mathbf{wm_2}$, even 90% weights are pruned, the watermark $\mathbf{wm_2}$ embedded in the model can still be successfully extracted. The reason is that, in model pruning attack, the weights with small values are set to be 0, therefore, a watermark with large values will not be pruned. In fact, when 90% weights are pruned, only the weights corresponding/map to watermark digits 0, 1, 2 are pruned, while the embedded watermark $\mathbf{wm_2}$ is not affected. Therefore, even for a small DNN model, embedding a watermark that does not contain small values (e.g., 0, 1, 2) can still effectively resist pruning attack. Overall, the proposed *ActiveGuard* method is effective and robust against the pruning attack.

Model	Attack	Epoch	Test accuracy	V_{owner}
LeNet-5	None	50	99.15%	**success**
	Fine-tuning attack	30	99.53%	**success**
	Fine-tuning attack	50	99.53%	**success**
WRN	None	200	91.46%	**success**
	Fine-tuning attack	30	91.47%	**success**
	Fine-tuning attack	50	91.53%	**success**

Table 3: Test Accuracy and Ownership Verification of Watermarked DNN under Fine-Tuning Attack

4 Conclusion

This paper proposes an active DNN IP protection technique via adversarial example based user fingerprinting. It protects the IP of DNN in three aspects: active authorization control, users' fingerprints management and copyright verification. Most of the existing works are passive verification methods, while this work can provide active copyright protection and copyright management for DNN. Compared to existing few active DNN IP protection works, the proposed method

Pruning rate	LeNet-5 (embedded with $\mathbf{wm}_1$)		LeNet-5 (embedded with $\mathbf{wm}_2$)		WRN (embedded with $\mathbf{wm}_1$)	
	Test accuracy	V_{owner}	Test accuracy	V_{owner}	Test accuracy	V_{owner}
0%	99.15%	**success**	99.16%	**success**	91.46%	**success**
10%	99.15%	**success**	99.15%	**success**	91.40%	**success**
20%	99.15%	**success**	99.09%	**success**	91.30%	**success**
30%	99.07%	**success**	99.08%	**success**	90.99%	**success**
40%	98.97%	**success**	99.06%	**success**	91.06%	**success**
50%	98.89%	failure	99.00%	**success**	90.28%	**success**
60%	98.54%	failure	98.93%	**success**	89.87%	**success**
70%	95.80%	failure	98.75%	**success**	88.03%	**success**
80%	91.02%	failure	97.46%	**success**	80.40%	**success**
90%	68.38%	failure	85.36%	**success**	48.35%	**success**

* $\mathbf{wm}_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 2, 1, 0]$, $\mathbf{wm}_2 = [3, 8, 7, 6, 8, 7, 6, 9, 9, 4, 8, 6, 5]$

Table 4: Test Accuracy and Ownership Verification of Watermarked DNN under Pruning Attack

can achieve users' fingerprints management and ownership verification, while introduces lower overhead.

Acknowledgment

This work is supported by CCF-NSFOCUS Kun-Peng Scientific Research Fund (No. CCF-NSFOCUS 2021012).

References

Allen, D. M. 1971. Mean Square Error Prediction as a Criterion for Selecting Regression Variables. *Technometrics*, 13(3): 469–475.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 39–57.

Chakraborty, A.; Mondal, A.; and Srivastava, A. 2020. Hardware-Assisted Intellectual Property Protection of Deep Learning Models. In *Proceedings of the 57th ACM/IEEE Design Automation Conference*, 1–6.

Chen, H.; Rouhani, B. D.; Fu, C.; Zhao, J.; and Koushanfar, F. 2019. DeepMarks: A Secure Fingerprinting Framework for Digital Rights Management of Deep Learning Models. In *Proceedings of the International Conference on Multimedia Retrieval*, 105–113.

Chen, M.; and Wu, M. 2018. Protect Your Deep Neural Networks from Piracy. In *Proceedings of the IEEE International Workshop on Information Forensics and Security*, 1–7.

Chollet, F.; et al. 2015. Keras. https://keras.io.

Fan, L.; Ng, K.; and Chan, C. S. 2019. Rethinking Deep Neural Network Ownership Verification: Embedding Passports to Defeat Ambiguity Attacks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 4716–4725.

Han, S.; Pool, J.; Tran, J.; and Dally, W. J. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Proceedings of the Advances in Neural Information Processing Systems*, 1135–1143.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

LeCun, Y.; Cortes, C.; and Burges, C. J. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lin, N.; Chen, X.; Lu, H.; and Li, X. 2021. Chaotic Weights: A Novel Approach to Protect Intellectual Property of Deep Neural Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(7): 1327–1339.

Pittaras, N.; Markatopoulou, F.; Mezaris, V.; and Patras, I. 2017. Comparison of Fine-Tuning and Extension Strategies for Deep Convolutional Neural Networks. In *Proceedings of the 23rd International Conference on MultiMedia Modeling*, 102–114.

Rouhani, B. D.; Chen, H.; and Koushanfar, F. 2019. DeepSigns: An End-to-End Watermarking Framework for Ownership Protection of Deep Neural Networks. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*, 485–497.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the 3rd International Conference on Learning Representations*, 1–14.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding Watermarks into Deep Neural Networks. In *Proceedings of the ACM on International Conference on Multimedia Retrieval*, 269–277.

Xue, M.; Wang, J.; and Liu, W. 2021. DNN Intellectual Property Protection: Taxonomy, Attacks and Evaluations (Invited Paper). In *Proceedings of the Great Lakes Symposium on VLSI*, 455–460.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference*, 1–15.

Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M. P.; Huang, H.; and Molloy, I. 2018. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *Proceedings of the Asia Conference on Computer and Communications Security*, 159–172.