Fine-tuning Is Not Enough:
A Simple yet Effective Watermark Removal Attack for DNN Models

Shangwei Guo[1] , **Tianwei Zhang**[2*] , **Han Qiu**[3] , **Yi Zeng**[4] , **Tao Xiang**[1] and **Yang Liu**[2]

[1]College of Computer Science, Chongqing University, China
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3]Institute for Network Sciences and Cyberspace, Tsinghua University, China
[4]The Bradley Department of Electronic and Computer Engineering, Virginia Tech Blacksburg, USA
{swguo,txiang}@cqu.edu.cn, {tianwei.zhang,yangliu}@ntu.edu.sg, qiuhan@tsinghua.edu.cn,
yizeng@vt.edu

Abstract

Watermarking has become the tendency in protecting the intellectual property of DNN models. Recent works, from the adversary's perspective, attempted to subvert watermarking mechanisms by designing watermark removal attacks. However, these attacks mainly adopted sophisticated fine-tuning techniques, which have certain fatal drawbacks or unrealistic assumptions. In this paper, we propose a novel watermark removal attack from a different perspective. Instead of just fine-tuning the watermarked models, we design a simple yet powerful transformation algorithm by combining imperceptible pattern embedding and spatial-level transformations, which can effectively and blindly destroy the memorization of watermarked models to the watermark samples. We also introduce a lightweight fine-tuning strategy to preserve the model performance. Our solution requires much less resource or knowledge about the watermarking scheme than prior works. Extensive experimental results indicate that our attack can bypass state-of-the-art watermarking solutions with very high success rates. Based on our attack, we propose watermark augmentation techniques to enhance the robustness of existing watermarks.

1 Introduction

Watermarking, originally designed for digital media [Petitcolas *et al.*, 1999], has been recently applied to the protection of Deep Learning (DL) models. This technique enables the ownership verification of DL models by embedding watermarks into the Deep Neural Networks (DNNs). One promising approach is data-poisoning watermarking [Zhang *et al.*, 2018; Adi *et al.*, 2018; Fan *et al.*, 2019; Zhang *et al.*, 2020], which modifies the model to give pre-defined output on some carefully-crafted watermark samples. Then the owners with only black-box accesses to the suspicious model can extract such watermarks by querying the model.

Similar to digital watermarking, a satisfactory data-poisoning watermarking mechanism must satisfy two requirements. The first one is *functionality-preserving*: the embedded watermarks should not affect the performance of the target model on normal samples. The second is *robustness*, where the watermarks cannot be removed with common model transformation, e.g., fine-tuning, model compression, etc. Even if the adversary knows the target model is watermarked, he has no means to remove the watermarks if he does not know the details of watermark samples.

Whether existing watermarking solutions are practically robust is widely challenged. A quantity of works try to invalidate these watermarking mechanisms and can be classified into two categories. The first strategy is to *detect* the verification queries and manipulate the responses. Some works [Namba and Sakuma, 2019; Aiken *et al.*, 2020] applied the strategies of backdoor attack detection to watermark detection, as some watermark designs are based on the backdoor techniques. However, these solutions fail to defeat backdoor attacks (as well as watermarks) with complex patterns [Liu *et al.*, 2019a; Tan and Shokri, 2019]. They are not applicable to watermark schemes based on other techniques (e.g., adversarial examples, out-of-distribution samples) either.

Another strategy is to directly *remove* the watermarks via model transformation. Although most watermarking solutions claim to be robust against various model transformations (e.g., fine-tuning, compression), recent works [Chen *et al.*, 2019; Shafieinejad *et al.*, 2019; Liu *et al.*, 2020] attempted to break such statement by introducing advanced fine-tuning methods with data augmentation, sophisticated loss function, etc. Unfortunately, these works suffer from significant limitations. For instance, [Shafieinejad *et al.*, 2019] requires original training samples to fine-tune the watermarked model. [Chen *et al.*, 2019] needs to know the type of the watermark to adjust the fine-tuning parameters. These assumptions make the attacks less unrealistic or practical.

It seems not promising to just use model fine-tuning for effective watermark invalidation. As a result, we propose a new watermark removal attack from a different direction: inference sample transformation. The key insight of our solution is that *watermark samples are less robust than normal samples*. Hence, we can design a preprocessing function to com-

---

promise the verification output while not affecting the normal output. Specifically, our novel function PST consists of a series of transformations (scaling, embedding random imperceptible patterns, spatial-level transformations). We further fine-tune the watermarked model with unlabelled out-of-distribution data to maintain the model's performance.

We are the *first* to explore the possibility of inference sample transformation for watermark removal. Our solution has much fewer requirements with more reasonable assumptions than prior works: it does not need any samples from the original training set (as in [Shafieinejad *et al.*, 2019]) or even following the original distribution (as in [Chen *et al.*, 2019; Liu *et al.*, 2020]). It is unified and does not need to know the adopted watermark mechanism for hyperparameter adjustment (as in [Chen *et al.*, 2019]). Its effectiveness is also confirmed by extensive evaluations.

Given the severity and practicality of our attack, we further introduce a defense methodology to enhance the robustness of watermarks. Inspired by the data augmentation technique which can enhance the model performance and generalization, we propose to augment the watermark samples with various transformations during the embedding process. In this case, the verification results remain correct even the adversary performs different preprocessing over the inference samples. Evaluations indicate this strategy can significantly improve the robustness of existing watermarking mechanisms.

2 Background and Related Works

2.1 Watermarking DNN Models

Existing watermarking schemes for DNNs can be classified into two categories. The first category is parameter-embedding watermarking [Uchida *et al.*, 2017; Rouhani *et al.*, 2019], which embed watermarks into the parameters without decreasing the model's performance. Verification of such watermarks require white-box access to the target models, which may not be achievable in some scenarios.

We are more interested in the second category, data-poisoning watermarking [Le Merrer *et al.*, 2019; Zhang *et al.*, 2018; Adi *et al.*, 2018]. These solutions take a set of carefully crafted sample-label pairs as watermarks and embed their correlation into DL models during the training process. We formally define the data-poisoning DNN watermarking scheme and illustrate the necessary properties.

Definition 1. *A DNN watermarking scheme with data poisoning is a tuple of probabilistic polynomial time algorithms (**WMGen**, **Mark**, **Verify**), where*

- **WMGen** *generates a set of watermarks $W = \{(x_i, y_i)\}_{i=1}^n$, in which x_i is a secret watermark input and y_i is the corresponding verification label.*
- **Mark** *embeds the watermarks into a DL model f and outputs the watermarked model $\widehat{f}$ such that $\widehat{f}(x_i) = y_i$ for $\forall\, (x_i, y_i) \in W$.*
- **Verify** *sends $\{x_i\}_{i=1}^n$ to a DL model $\widetilde{f}$ and obtains the predictions $\{y_{\widetilde{f},i}\}_{i=1}^n$. If the probability that $y_{\widetilde{f},i}$ equals y_i for $i \in [1, n]$ is larger than a predefined value τ, **Verify** outputs 1. Otherwise it outputs 0.*

Let $P_{\widehat{f},D}$ be the prediction accuracy of $\widehat{f}$ on the normal samples following the data distribution D; $P_{\widehat{f},W}$ be the accuracy of $\widehat{f}$ on the verification watermarks W, i.e.,

$$P_{\widehat{f},D} = Pr(\widehat{f}(x) == f(x), x \backsim D), \qquad (1)$$

$$P_{\widehat{f},W} = Pr(\widehat{f}(x_i) == y_i, \{x_i, y_i\} \in W). \qquad (2)$$

A data-poisoning watermarking scheme should have the following basic properties: (1) *Functionality-preserving*: $P_{\widehat{f},D}$ should be identical with $P_{f,D}$. (2) *Robustness*: **Verify** outputs 1 even if $\widehat{f}$ is slightly transformed to a different model $\widetilde{f}$ using common techniques such as fine-tune or model compression, i.e. $P_{\widetilde{f},W} > \tau$.

Various state-of-the-art data-poisoning techniques have been proposed to embed watermarks into DNN models. These methods can be classified into three categories:

- **Perturbation-based**: the samples are generated by slightly perturbing normal samples, where the difference between the two types of samples is bounded. They can be generated using the adversarial frontier stitching [Le Merrer *et al.*, 2019] or GAN-based techniques [Li *et al.*, 2019].
- **Pattern-based**: the samples are generated by embedding a certain pattern into normal images using backdoor techniques [Adi *et al.*, 2018; Zhang *et al.*, 2018].
- **Out-of-distribution (OOD)**: the samples are randomly selected from the internet and totally different from the normal ones [Adi *et al.*, 2018].

2.2 Attacking Watermarking Schemes

We consider a threat model where the adversary has white-box accesses to the target model and is able to alter the model. He aims to break the property of Equation (2) but still maintain the property of Equation (1). A good removal attack must meet the following requirements.

- *Comprehensive*: the attack should be able to invalidate all existing state-of-the-arts watermarking solutions, including perturbation-based, pattern-based, and OOD.
- *Watermark-agnostic*: the adversary has no knowledge about the employed watermarking scheme, including the technique and watermark type. He has no access to the original training samples, or the distribution.
- *Efficient*: the adversary should be able to remove the watermarks in a lightweight manner, i.e., with much less computation cost than training a model from scratch. Otherwise, he will lose the motivation of stealing the target model, and just train his own copy.

Existing watermark attacks fail to satisfy all these requirements. These solutions can be classified into two categories. The first approach is to distinguish the watermark samples from normal samples during the inference process, and then alter the verification results [Namba and Sakuma, 2019; Aiken *et al.*, 2020]. However, these attacks are not watermark-agnostic, as they require the entire training dataset. They are not comprehensive as they only work for simple pattern-based watermarks.
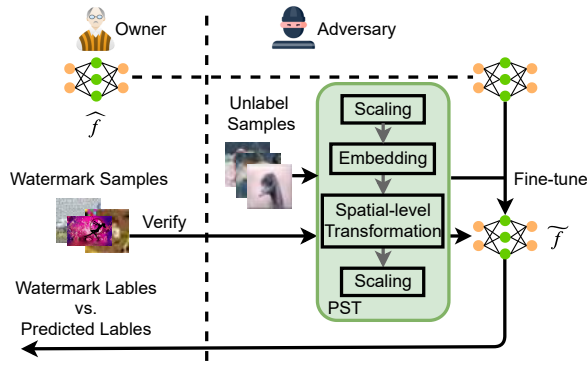
Figure 1: Overview of the proposed attack.

The second is to remove watermarks from the model using fine-tuning techniques [Chen *et al.*, 2019; Shafieinejad *et al.*, 2019; Liu *et al.*, 2020]. They carefully designed learning rate schedule and synthesized fine-tuning samples to make the watermarked model forget the watermark samples. Unfortunately, these approaches also suffers from several fatal limitations: they are not watermark-agnostic, as they need to access the original training data [Shafieinejad *et al.*, 2019; Chen *et al.*, 2019; Liu *et al.*, 2020]. [Chen *et al.*, 2019] also needs to know the watermark type for hyperparameter adjustment. [Liu *et al.*, 2020] is not comprehensive to cover the OOD watermarking scheme.

3 Proposed Attack

Figure 1 illustrates the workflow of our proposed attack. The core of our solution is a powerful function, **P**attern embedding and **S**patial-level **T**ransformation (PST). Given a watermarked model $\widehat{f}$, the adversary first preprocesses some OOD data samples with PST, and uses them to fine-tune $\widetilde{f}$ for a few epochs. Then he can serve the fine-tuned model $\widetilde{f}$ online. During inference, he applies PST for each sample and feeds it to $\widetilde{f}$. The model output of a normal sample will remain the same while the watermark ones will be altered.

Key Insight. Our approach is based on the robustness gap between watermark and normal samples. Intuitively, a data-poisoning watermarked model is trained to memorize the relationships between $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$. During the verification, the watermarked model will predict y_i given x_i. However, such memory may be fragile because y_i has to be different from the ground truth of x_i. Based on this, PST is expected to preprocess watermark samples, and make them unrecognizable by $\widehat{f}$. To maintain the model's performance over normal samples, we leverage PST to preprocess some OOD samples and fine-tune the watermarked model.

3.1 PST

Our PST function consists of three modules: (1) *scaling* is to resize an input sample to a fixed size; (2) *imperceptible pattern embedding* introduces human-unnoticeable perturbations to reduce the impact of abnormal pixels and patterns in the watermark samples; (3) *spatial-level transformations*

further destroy the special spatial shapes introduced by the watermarking scheme. Detailed steps are illustrated below.

Step 1: Scaling. Given a sample x of an arbitrary size, PST first scales it to a fixed size. Specifically, PST chooses the bicubic interpolation [Meijering, 2002] for scaling to reduce the correlation between watermark samples and labels. This operation `Bicubic` interpolates pixels by considering a small square neighborhood (4×4 in our experiments): $x = \texttt{Bicubic}(x, \beta)$, where β is the scaling paramater. We choose this bicubic interpolation because samples with bicubic interpolation have fewer interpolation artifacts and often outperform the ones with bilinear or nearest-neighbor interpolation. Thus, this step can better preserve the performance of $\widehat{f}$ on normal samples.

Step 2: Imperceptible Pattern Embedding. Besides the bicubic interpolation, PST embeds random imperceptible patterns into the scaled watermark samples to further affect the prediction of $\widehat{f}$. Due to the imperceptibility property, the embedded pattern will not impact the performance of $\widehat{f}$ over normal samples. Since the adversary has no prior knowledge about the watermarking scheme and watermark samples, we design a random yet easily identifiable pattern to alter the memory of $\widehat{f}$ about the watermark samples.

In particular, we adopt a random grid median filter `MedianFilter` to generate the desired imperceptible pattern. During the embedding, PST first selects a set of random rows and columns with the same interval size v. For each pixel $x_{i,j}$ in the selected rows and columns, it is replaced by the pixel with the median value in the neighborhood. The pattern size is controlled by the interval size v: a large v can lead to a small number of the selected columns and rows, and a more imperceptible pattern. This median filter can be replaced by other types of filters such as the maximum filter.

Step 3: Spatial-level Transformations. Although the above pixel-level transformations can affect the watermark samples with small perturbations, they are insufficient to remove large perturbations such as pattern-based watermarks. To this end, we propose to adopt both linear and nonlinear spatial-level transformations to further compromise the effects of watermark samples. We integrate two different transformations and use a parameter γ to uniformly control the strength of each one. Specifically, Let (i, j) be a pixel of x and $(\widetilde{i}, \widetilde{j})$ be the corresponding transformed pixel. For each operation, we require the distance between each dimension of the input and output pixels to be lower than γ, i.e.,

$$|i - \widetilde{i}| \le \gamma, \text{ and } |j - \widetilde{j}| \le \gamma.$$

Our PST first adopts random affine transformations (`Affine`) over the pattern-embedded x. An affine transformation is linear and can be formalized as a 2×3 matrix, which preserves points, straight lines, and planes of samples. For a pixel (i, j), it calculates

$$\begin{bmatrix} \widetilde{i} \\ \widetilde{j} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{bmatrix} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}. \tag{3}$$

For example, the matrix of the translation transformation in the horizontal direction is $\begin{bmatrix} 1 & 0 & \phi \\ 0 & 1 & 0 \end{bmatrix}$. To meet the distance constraint, we randomly choose ϕ such that $|\phi| \leq \gamma$ for all the columns. We can combine multiple affine transformations together such as rotation and translation.

Besides affine transformations, we also adopt an elastic transformation `Elastic` to further modify the samples, which is a nonlinear transformation that produces random elastic distortions. It first generates random displacement fields that are convolved with a Gaussian distribution of standard deviation σ. The displacement fields are then multiplied by a scaling factor α that controls the intensity of the deformation. We adjust the standard deviation σ and the scaling factor α to restrict the displacement to be smaller than γ.

3.2 Attack Pipeline

Algorithm 1 describes the details of the two-stage attack:

Offline fine-tuning (Lines 17-21). The adversary first adopts PST to fine-tune $\widehat{f}$ to enhance the model performance. Different from prior works [Chen *et al.*, 2019; Shafieinejad *et al.*, 2019; Liu *et al.*, 2020] which require the original labeled training data or distribution, our method can select an arbitrary set of unlabeled and out-of-distribution samples $\{x_i'\}$. The adversary first calculates the corresponding label of each sample using $\widehat{f}$ as an oracle. Then he establishes a dataset $\{(g^{pst}(x_i'), \widehat{f}(g^{pst}(x_i')))\}$ to fine-tune the model $\widehat{f}$ for a few epochs to reach the desired performance. This fine-tuning process takes very short time to complete, and can effectively adjust the noise and perturbations introduced by PST.

Online inference (Lines 1-7). For each sample x, the online model returns $\widetilde{y} = \widetilde{f}(g^{pst}(x))$. When the owner executes the ***Verify*** function to query the model with a watermark sample x_i, $\widetilde{y}_i$ is different from y_i with a high probability, due to the effectiveness of g^{pst} and $\widetilde{f}$. Thus, the similarity between $\{y_i\}_{i=1}^n$ and $\{\widetilde{y}_i\}_{i=1}^n$ would be smaller than τ, which can prevent the model owner from identifying the watermarks. In contrast, $\widetilde{f}$ will give the correct results for normal samples x.

4 Experiments

4.1 Experimental Setup

Datasets and DNN models. Our attack is model-agnostic and dataset-agnostic. Following existing works, we conduct experiments on the CIFAR datasets with the same number of training samples. We evaluate our attack on the ResNet model [He *et al.*, 2016]. We adopt a batch size of 100 and the learning rate decay with an initialize value of 0.2.

Watermark samples. We evaluate our attack on four types of state-of-the-art watermark samples: 1) Perturbation: the watermark samples are generated by adding imperceptible perturbations on normal samples. They are close to the frontiers of the models [Le Merrer *et al.*, 2019]. 2) Patch: the watermark samples are created by embedding a specific patch into the normal samples [Zhang *et al.*, 2018]. 3) Content:

Algorithm 1: Our watermark removal attack.

Parameters: $\beta, v, \gamma, \widehat{f}, \{x_i'\}, \{x_i, y_i\}_{i=1}^n$

1 **Function *Verify* ($\{x_i, y_i\}_{i=1}^n$):**
2   **foreach** $x_i \in \{x_i\}_{i=1}^n$ **do**
3       $\widetilde{y}_i \leftarrow \widetilde{f}(g^{pst}(x_i))$, where $\widetilde{f} \leftarrow$ `Fine-tune`;
4   **if** $d(\{y_i\}_{i=1}^n, \{\widetilde{y}_i\}_{i=1}^n) < \tau$ **then**
5   **return** 1
6   **else**
7   **return** 0

8 **Function g^{pst} (x):**
9    $x \leftarrow$ `Bicubic`(x, β);
10   Randomly selected rows and columns with interval v;
11   **foreach** $x_{i,j}$ in the selected rows and columns **do**
12       $x_{i,j} \leftarrow$ `MedianFilter`$(x_{i,j})$;
13    $x \leftarrow$ `Affine`(x, γ);
14    $x \leftarrow$ `Elastic`(x, γ);
15    $\widetilde{x} \leftarrow$ `Bicubic`$(x, \frac{1}{\beta})$;
16   **return** $\widetilde{x}$

17 **Function `Fine-tune`($\widehat{f}, \{x_i'\}$):**
18   **foreach** $x_i' \in \{x_i'\}$ **do**
19       $y_i' \leftarrow \widehat{f}(g^{pst}(x_i'))$;
20    $\widetilde{f} \leftarrow$ `Train`$(\widehat{f}, \{g^{pst}(x_i'), y_i'\})$;
21   **return** $\widetilde{f}$

---

the watermark samples are generated by embedding the word "TEXT" into the normal samples [Zhang *et al.*, 2018]. 4) OOD: the watermark samples are randomly selected from unrelated datasets [Adi *et al.*, 2018].

Figure 2 (first row) illustrates an example of a normal sample and the corresponding four types of watermark samples. Following the implementation in [Adi *et al.*, 2018; Le Merrer *et al.*, 2019], we set the number of watermark samples as 100. We set τ as 60% for both CIFAR10 and CIFAR100 when perturbation-based watermark samples are involved. Otherwise, τ is 20% for CIFAR10 and 10% for CIFAR100. We train models from scratch to embed the watermarks. Each training process is stopped after 60 epochs if the model accuracy on watermark samples is 1. Otherwise, we continue the training till all watermarks are embedded.

Implementation Details. We *uniformly* set the parameters for all datasets and watermarks, which indicates our attack does not require any prior knowledge of the watermarking mechanisms. In particular, for PST, we set β as 5 and the size of the scaled images is 160×160. We set the interval size as 5 to generate the imperceptible pattern. We set the displacement bound γ in spatial-level transformations as 15. We use both in-distribution and out-of-distribution unlabeled samples in the fine-tuning phase. Specifically, the in-distribution samples are generated from the corresponding datasets using data augmentation techniques. The out-of-distribution samples of CIFAR10 (resp. CIFAR100) are randomly selected from CIFAR100 (resp. CIFAR10). We use 25000 unlabeled samples and fine-tune the watermarked models for 10 epochs.

Figure 2: Examples of the watermark samples in our experiments. First row: normal sample and the four types of watermark samples; Second row: the samples preprocessed using PST with $\gamma = 10$; Third row: the samples preprocessed using PST with $\gamma = 20$.

Datasets	Attack	Perturbation		Patch		Content		OOD	
		P_D	P_W	P_D	P_W	P_D	P_W	P_D	P_W
CIFAR10	None	0.92	1.00	0.93	1.00	0.94	1.00	0.92	1.00
	BdR	0.45	**0.36**	0.48	1.00	0.48	0.89	0.47	**0.11**
	Shield	0.67	**0.34**	0.63	1.00	0.61	0.97	0.66	**0.11**
	PD	0.51	**0.47**	0.56	0.94	0.54	0.80	0.53	**0.11**
	FD	0.57	**0.61**	0.59	1.00	0.59	0.53	0.55	**0.15**
	GB	0.54	**0.46**	0.66	0.98	0.62	0.06	0.66	**0.15**
	ST	0.83	**0.46**	0.86	0.11	0.84	0.11	0.83	0.06
	PST	0.82	**0.43**	0.85	0.09	0.84	0.11	0.82	0.06
	FT-In	**0.91**	1.00	**0.92**	0.94	**0.92**	1.00	**0.91**	0.70
	FT-Out	**0.89**	0.97	**0.90**	0.59	**0.91**	0.79	**0.89**	0.71
	Proposed_In	**0.89**	0.34	**0.91**	0.01	**0.91**	0.05	**0.89**	0.04
	Proposed_Out	**0.87**	0.47	**0.89**	0.03	**0.89**	0.06	**0.87**	0.08
CIFAR100	None	0.74	1.00	0.74	1.00	0.75	1.00	0.73	1.00
	BdR	0.16	**0.05**	0.14	0.79	0.16	0.56	0.15	**0.00**
	Shield	0.30	**0.15**	0.30	0.97	0.29	0.61	0.32	**0.02**
	PD	0.25	**0.32**	0.24	0.44	0.25	0.41	0.23	**0.01**
	FD	0.33	**0.20**	0.33	0.27	0.33	0.31	0.32	**0.03**
	GB	0.39	**0.13**	0.42	0.91	0.41	0.01	0.43	**0.02**
	ST	0.60	**0.20**	0.61	0.20	0.61	0.08	0.61	**0.01**
	PST	**0.69**	0.20	**0.69**	0.20	**0.69**	0.07	0.68	**0.00**
	FT-In	**0.71**	0.93	**0.71**	1.00	**0.70**	0.95	**0.71**	0.85
	FT-Out	0.66	0.73	0.67	0.74	0.67	0.88	0.66	0.75
	Proposed_In	**0.73**	0.18	**0.74**	0.03	**0.73**	0.05	**0.73**	0.00
	Proposed_Out	**0.69**	0.16	**0.70**	0.05	**0.69**	0.07	**0.69**	0.00

Table 1: Experimental results of watermark removal attacks on normal samples and watermark samples.

4.2 Effectiveness of Our Attack

We consider two strategies: the first one is to just use input preprocessing to remove watermarks. The attacker can choose PST transformation, or PST without imperceptible pattern embedding (denoted as ST). For comparisons, we also implement some state-of-the-art transformations for adversarial example mitigation as baselines: BdR [Xu *et al.*, 2017], Shield [Das *et al.*, 2018], PD [Prakash *et al.*, 2018], FD [Liu *et al.*, 2019b], and GB [Hendrycks and Dietterich, 2019]. Since they are proposed to remove small perturbations in adversarial examples, we adjust the parameters of these methods to affect samples with large patterns till their accuracy is lower than 70% (resp. 50%) on CIFAR10 (resp. CIFAR100).

The second strategy is the end-to-end attack with both PST and fine-tuning. We compare it with the black-box watermark removal attack (FT) in [Shafieinejad *et al.*, 2019] which only fine-tunes the watermarked models with unlabeled samples. FT takes a set of unlabeled samples as input and obtains their labels using the watermarked model as an oracle. Then, it fine-tunes the watermarked models for 10 epochs with the new sample-label pairs. To be comprehensive, we implement the two solutions with in-distribution data for fine-tuning as well (Proposed-In and FT-In).

Since these input transformation functions introduce high randomization, we run each experiment five times to obtain the statistical results. We calculate the average performance of the attacks on normal samples P_D to reflect their usability. We select the minimum accuracy of the attacks on watermark samples P_W to quantify their effectiveness, which is efficient to reflect the threat an attack can bring. The experimental results lead to the same conclusion with the average accuracy.

The experimental results are illustrated in Table 1. For the first strategy, we observe that perturbation and OOD watermarks are vulnerable to all kinds of transformations. The adversarial example transformations (except FD) and PST (also ST) can easily remove these two types of watermarks on both CIFAR10 and CIFAR100. However, adversarial example transformations are not helpful in removing pattern-based watermarks. Although GB can remove Content watermarks on CIFAR10 and CIFAR100, all adversarial example trans-

formations can not affect Patch watermarks even P_D is extremely low. In contrast, PST can significantly reduce the accuracy of the watermarked model on all types of watermarks without heavily affecting P_D. ST is slightly worse than PST, but is still much better than the other transformations.

For the second strategy, we observe that although FT-In and FT-Out have almost the same performance on normal samples, they cannot remove watermarks, which was confirmed by the results in existing watermarking papers [Adi *et al.*, 2018; Zhang *et al.*, 2018]. Our attack (Proposed-In and Proposed-Out) can remove all watermarks and preserve acceptable P_D values. Besides, compared with PST, the proposed fine-tuning process can not only improve the performance of the models on normal samples, but also further cleanse the watermarks.

4.3 Impact of Parameters

Varying the number of unlabeled samples. We evaluate the performance of our attack with different numbers of unlabeled samples on CIFAR10. Figure 3 shows the performance of our attack on Patch watermarks. We observe that there exists a trade-off: more fine-tuning samples can increase the performance, which is harder to obtain. If the adversary has enough unlabeled samples, he can obtain a high-quality model with the same performance as the watermarked one. Beside, the success rates of our attack are preserved and even slightly increases with more unlabeled samples, which indicates its severe threat to existing watermarking schemes.

Varying γ. Figure 2 visually illustrates the transformed output of normal and watermark samples with $\gamma = 10$ (second row) and $\gamma = 20$ (third row). Two observations are made from this figure. First, the distortion caused by PST increases as the displacement bound γ increases. Second, the samples are randomly preprocessed, which indicates that our attack can be adopted for various watermarking schemes with different settings. We also measure the performance of our at-
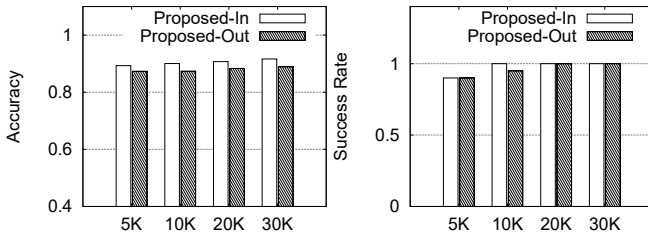
Figure 3: Performance of our attack with different numbers of unlabeled in-distribution and out-of-distribution samples.
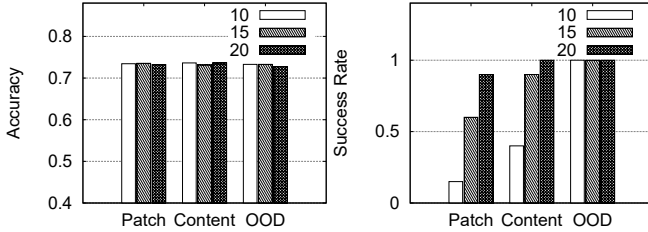

Figure 4: Performance of Proposed-Out with different γ values on CIFAR100.

Datasets	Augmentation	Perturbation		Patch		Content		OOD	
		P_D	P_W	P_D	P_W	P_D	P_W	P_D	P_W
CIFAR10	ShearX	0.84	0.90	0.86	0.84	0.84	0.58	0.85	0.81
	Rotate	0.84	0.85	0.84	0.23	0.84	0.15	0.85	0.73
	PST	0.80	1.00	0.79	1.00	0.81	1.00	0.84	0.99
CIFAR100	ShearX	0.68	0.63	0.70	0.70	0.68	0.43	0.69	0.77
	Rotate	0.68	0.64	0.68	0.20	0.69	0.21	0.68	0.62
	PST	0.69	0.99	0.68	0.94	0.66	0.96	0.70	1.00

Table 2: Experimental results of watermark augmentation defenses against our attack on normal and watermark samples.

tack with different γ values on CIFAR100. The experimental results are presented in Figure 4. We observe that due to the effect of the distortion, the model accuracy decreases (except Content) and the success rate increase on the four types of watermarks with an increased γ.

5 Watermark Augmentation for Robustness Enhancement

As discussed in previous sections, inference preprocessing with model fine-tuning can effectively remove state-of-the-art watermarks from the protected model. So we aim to seek for new solutions to enhance the robustness of existing watermarking schemes. Inspired by the data augmentation techniques that can improve the generalization ability of DNN models, we propose *watermark augmentation* to strengthen the watermarks during the embedding process.

Methodology. A variety of transformations [Cubuk *et al.*, 2019] have been designed for data augmentation. Given a specific watermarking scheme with a set of watermarks, we apply those transformations over each watermark sample to augment the watermark set. Then we follow the ***Mark*** procedure to embed the original and transformed samples into the model. With this watermark augmentation method, the protected model is expected to recognize the watermark samples even the model and samples are perturbed by the adversary.

Evaluations. We leverage three augmentation transformations: ShearX, Rotate, and PST. Each function has different strength values to transform images. At each iteration, we randomly sample an augmentation strength value and apply the selected transformation to the watermark samples, before embedding them to the model.

We measure the effectiveness of the watermark augmentation strategy against our PST on CIFAR10 and CIFAR100. Table 2 illustrates the experimental results of the three augmentation defenses. We observe that watermark augmenta-

tion can significantly improve the robustness of the watermarks (expect Rotate on CIFAR10) compared to the results without any defense in Table 1. Besides, watermark augmentation with PST is more effective than other transformations.

6 Conclusion and Future Work

In this paper, we propose an efficient and practical attack to blindly remove watermarks in DL models. We introduce a novel preprocessing function PST, to transform samples for model fine-tuning and preprocess inference samples. Extensive experiments demonstrate that the proposed attack technique can remove various types of state-of-the-art watermarks without any prior knowledge about the watermarking schemes and labeled training samples. This indicates existing watermarking schemes still have high severity vulnerabilities. We also propose a new defense strategy to enhance the robustness of existing watermark mechanisms.

For the attack, we empirically and comprehensively validate our technique, following all existing relevant works. Unfortunately, we cannot formally prove the effectiveness of our solutions against all watermarking schemes. Formal and theoretical analysis about watermarks and attacks is an open and challenging problem. To our best knowledge, currently there are no works or attempts to address this problem yet. We will consider this as an important direction for future work. We will also evaluate our attack as a defense against adversarial examples and backdoors.

For the defense, evaluations show the model with augmented watermarks is more robust against simple attacks. The possibility of adaptive attacks targeting this defense is unknown and worth exploration. We hope our work can heat up the arms race between the model owner and plagiarist to inspire the designs of more advanced watermark schemes and attacks in the future.

Acknowledgments

References

[Adi *et al.*, 2018] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security Symposium*, pages 1615–1631, 2018.

[Aiken *et al.*, 2020] William Aiken, Hyoungshick Kim, and Simon Woo. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *arXiv preprint arXiv:2004.11368*, 2020.

[Chen *et al.*, 2019] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. REFIT: A unified watermark removal framework for deep learning systems with limited data. *arXiv preprint arXiv:1911.07205*, 2019.

[Cubuk *et al.*, 2019] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. 2019.

[Das *et al.*, 2018] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using JPEG compression. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 196–204, 2018.

[Fan *et al.*, 2019] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *Advances in Neural Information Processing Systems*, pages 4714–4723, 2019.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[Hendrycks and Dietterich, 2019] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[Le Merrer *et al.*, 2019] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, pages 1–12, 2019.

[Li *et al.*, 2019] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In *Annual Computer Security Applications Conference*, pages 126–137, 2019.

[Liu *et al.*, 2019a] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

[Liu *et al.*, 2019b] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature distillation: DNN-oriented JPEG compression against adversarial examples. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 860–868, 2019.

[Liu *et al.*, 2020] Xuankai Liu, Fengting Li, Bihan Wen, and Qi Li. Removing backdoor-based watermarks in neural networks with limited data. *arXiv preprint arXiv:2008.00407*, 2020.

[Meijering, 2002] Erik Meijering. A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, 90(3):319–342, 2002.

[Namba and Sakuma, 2019] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *ACM Asia Conference on Computer and Communications Security*, pages 228–240, 2019.

[Petitcolas *et al.*, 1999] Fabien AP Petitcolas, Ross J Anderson, and Markus G Kuhn. Information hiding-a survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999.

[Prakash *et al.*, 2018] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8571–8580, 2018.

[Rouhani *et al.*, 2019] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for protecting the ownership of deep neural networks. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.

[Shafieinejad *et al.*, 2019] Masoumeh Shafieinejad, Jiaqi Wang, Nils Lukas, Xinda Li, and Florian Kerschbaum. On the robustness of the backdoor-based watermarking in deep neural networks. *arXiv preprint arXiv:1906.07745*, 2019.

[Tan and Shokri, 2019] Te Juin Lester Tan and Reza Shokri. Bypassing backdoor detection algorithms in deep learning. *arXiv preprint arXiv:1905.13409*, 2019.

[Uchida *et al.*, 2017] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *ACM on International Conference on Multimedia Retrieval*, pages 269–277, 2017.

[Xu *et al.*, 2017] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[Zhang *et al.*, 2018] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *ACM Asia Conference on Computer and Communications Security*, pages 159–172, 2018.

[Zhang *et al.*, 2020] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 12805–12812, 2020.