Evaluating the Robustness of Trigger Set-Based Watermarks Embedded in Deep Neural Networks

Suyoung Lee, Wonho Song, Suman Jana, Meeyoung Cha, and Sooel Son, Member, IEEE

Abstract—Trigger set-based watermarking schemes have gained emerging attention as they provide a means to prove ownership for deep neural network model owners. In this paper, we argue that state-of-the-art trigger set-based watermarking algorithms do not achieve their designed goal of proving ownership. We posit that this impaired capability stems from two common experimental flaws that the existing research practice has committed when evaluating the robustness of watermarking algorithms: (1) incomplete adversarial evaluation and (2) overlooked adaptive attacks. We conduct a comprehensive adversarial evaluation of 11 representative watermarking schemes against six of the existing attacks and demonstrate that each of these watermarking schemes lacks robustness against at least two non-adaptive attacks. We also propose novel adaptive attacks that harness the adversary’s knowledge of the underlying watermarking algorithm of a target model. We demonstrate that the proposed attacks effectively break all of the 11 watermarking schemes, consequently allowing adversaries to obscure the ownership of any watermarked model. We encourage follow-up studies to consider our guidelines when evaluating the robustness of their watermarking schemes via conducting comprehensive adversarial evaluation that includes our adaptive attacks to demonstrate a meaningful upper bound of watermark robustness.

Index Terms—Deep neural networks, watermark removal attacks, backdoor attacks, watermark robustness, trigger set-based watermarks

1 INTRODUCTION

THE recent advent of deep neural networks (DNNs) has accelerated the development and application of diverse DNN models across various domains, including image search [19], [53], security [56], [57], and self-driving vehicles [50]. As machine learning technology evolves, the structures of state-of-the-art DNN models have become more complicated. This trend renders corporations with fewer computational resources unable to train state-of-the-art DNN models from scratch. For instance, the ImageNet [43] dataset holds 14M images; training a high-performing DNN model such as a ResNet-50 [22], which consists of over 25M parameters, takes up to several weeks with a machine equipped a Tesla M40 GPU. Moreover, it is difficult to obtain a large number of high-quality training instances pertaining to privacy-sensitive information, thus rendering it infeasible for corporations with limited data access to produce a superb model.

An adversary may attempt to steal such a superb model and host another service that imitates the service provided by the original model. This adversary poses a grave threat to the model owner, who has invested resources and time to develop a high-performing model. DNN model theft thus infringes on the intellectual property (IP) of the model owner and discloses the owner’s business secrets. Accordingly, corporations seek a mechanism that proves the ownership of their DNN models to protect their IPs and business secrets.

Previous studies have proposed novel methods that validate ownership of a given DNN model, thus protecting the owner’s IP. Similar to watermarking algorithms devised to protect the IP of multimedia content, such as images and videos [28], [47], previous studies have proposed new ways of embedding watermarks into a given DNN model as well as algorithms that verify ownership [2], [10], [16], [21], [24], [32], [36], [37], [42], [52], [59], [60]. The proposed watermarking algorithms are categorized into two types based on their methods of embedding watermarks: feature-based and trigger set-based methods.

Feature-based schemes [10], [42], [52] require white-box access to a model’s internal weight parameters. On the other hand, trigger set-based watermarking methods [2], [16], [21], [24], [32], [36], [37], [59], [60] have gained attention due to their comparative merits of requiring black-box access for ownership verification. Trigger set-based schemes harness the common query interface of a suspect model. Specifically, these watermarking methods leverage carefully created images, called key images. A model owner assigns an arbitrary label, called a target label, to the key images and generates a trigger set that consists of an arbitrary number of key image and target label pairs. The owner then trains a model on this trigger set as well as on the normal training data. When verifying ownership, the model owner queries the model in doubt with the key images and checks whether the model returns the target label; this enables the owner to verify the ownership by using remote queries. Previous trigger set-based methods [2], [16], [21], [32], [36], [37], [59], [60]
have in common that they use key images and target labels but differ in how they generate key images or select target labels.

**Contributions.** In this paper, we argue that trigger set-based watermarking methods today [2], [16], [21], [24], [32], [36], [37], [59], [60] do not achieve their goal of enabling model owners to prove their ownership of watermarked DNN models. We believe that previous studies have not evaluated the robustness of their watermarking algorithms to the fullest extent, thereby failing to demonstrate their readiness for real-world deployment.

There exist two different strategies for evaluating the robustness of a DNN model: (1) proving a theoretical lower bound with approximation [3], [23] and (2) demonstrating an upper bound via adversarial evaluation with strong attacks. Previous watermarking studies [2], [16], [21], [24], [32], [36], [37], [59], [60] have taken the latter approach, demonstrating their robustness against selected attacks. However, we observed two common flaws in previous studies when evaluating the robustness of their trigger set-based watermarking algorithms: (1) performing incomplete adversarial evaluation and (2) overlooking an adaptive adversary.

**Incomplete adversarial evaluation.** Because various attacks have been introduced across diverse studies in various contexts, we first consolidate and reorganize six existing attacks. We then categorize the existing attacks into two types, each of which corresponds to either of the adversary’s two strategies: (1) claiming ownership by the adversary or (2) obscuring the owner’s ownership.

We observed that no previous watermarking studies have considered the complete set of the existing strong attacks in their adversarial evaluation; previous studies have not demonstrated their robustness against at least one critical attack. Furthermore, we contend that adversarial evaluation of attacks employing the adversary’s first strategy (claiming ownership by the adversary) is unnecessary. This strategy permits for a target model to contain watermarks from its original owner as well as the adversary, which demands additional proof to prove the adversary’s ownership (§5.3.1). Therefore, there is no motive for the adversary to employ this strategy alone unless she combines the two aforementioned strategies by obscuring the original watermarks and then injecting her own watermarks.

To this end, we perform our own adversarial evaluation against 11 of the representative trigger set-based watermarking schemes while taking into account the aforementioned problems. We demonstrate that they are weak against at least two of these attacks. In particular, all of the 11 evaluated schemes were vulnerable to evasion [32], [35], [37] and ownership piracy attacks [2].

**Overlooked adaptive adversary.** Previous studies focused on evaluating their watermark robustness against selected existing attacks. Meanwhile, a vast volume of recent research on establishing the robustness of DNN models has considered adaptive adversaries [4], [6], [13], [25], [41], [46].

To this end, we propose three novel attacks that a strong adaptive adversary is able to conduct. Under the assumption that this adversary knows the underlying watermarking algorithm of a target model, we demonstrate that the proposed adaptive attacks effectively break all existing watermarking schemes, enabling the adversary to obscure the ownership of a target model, regardless of its underlying watermarking scheme. Therefore, our proposed attacks contribute to demonstrating a new upper bound of watermark robustness.

Overall, our experimental results demonstrate that trigger set-based watermarking schemes today are far from ready for real-world deployment. We recommend that future research evaluate their watermarking methods against at least all existing strong attacks, including our adaptive attacks, and consider our guidelines when demonstrating their watermark robustness via adversarial evaluation (§8).

To enable follow-on research to evaluate its watermarking schemes, all of our attack algorithms and their implementation will be available at https://github.com/WSP-LAB/wm-eval-zoo.

## 2 Background

### 2.1 DNN Ownership Verification

Since Uchida et al. [52] proposed the first approach to embedding watermarks into neural networks, various watermarking techniques have been proposed. In terms of their watermark embedding methodology, these watermarking techniques have been categorized into two types: trigger set-based and feature-based methods. Trigger set-based methods utilize additional training samples as watermarks for DNNs [2], [16], [21], [24], [32], [36], [37], [59], [60]. Feature-based methods embed watermarks by modifying model features [10], [42], [52].

Zhang et al. [59] proposed a representative trigger set-based method. They trained a model to learn predefined key pairs, each consisting of a key image and its target label. Specifically, they assigned a false label with respect to the ground-truth function to the key image. The gist of their approach is that a model without the watermark is highly likely to emit a ground-truth label rather than the predefined false label for a given key image. Therefore, the owner can prove the ownership afterward by querying the model with the key images and checking whether the model outputs the predefined false label. In this scheme, the key images and their predefined false labels become a trigger set.

Other trigger set-based watermarking techniques employ more or less similar approaches, but Adi et al. [2] further integrated this scheme with cryptographic primitives to secure embedded watermarks. Recently, Jia et al. [24] proposed to train a model in the direction of tightly coupling the trigger set with a regular training set so that the trained model becomes robust against model stealing attacks.

### 2.2 Target Watermark Schemes

Our goal is to evaluate the robustness of state-of-the-art trigger set-based watermarking schemes. Thus, we chose 11 representative watermarking algorithms, published at top venues over the past five years [2], [16], [21], [24], [32], [36], [37], [42], [59]. They share a common scheme that uses trigger sets for verifying ownership.

Algorithm 1 summarizes how a trigger set-based watermark algorithm embeds the ownership proof of an owner $O$ into a DNN model. $O$ provides a training set $D_{\text{train}}$ and
Algorithm 1: Embedding a trigger set into a DNN.

```
Input: A regular training set ($D_{train}$).
A set of source images ($I_{src}$).

Output: A watermarked model ($M_{wm}$).

function EmbedWatermark($D_{train}$, $I_{src}$)

1. $I_{key} \leftarrow \text{GenerateKeyImgs($I_{src}$)}$
2. $L_{target} \leftarrow \text{AssignTargetLabels($I_{key}$)}$
3. $D_{trigger} \leftarrow \text{AssignKeyLabels($I_{key}$)}$
4. $M_{wm} \leftarrow \text{TrainModel($D_{train}$, $D_{trigger}$)}$

return $M_{wm}$
```

a set of source images $I_{src}$ to the EmbedWatermark function. Given $I_{src}$, GenerateKeyImgs generates a set of key images $I_{key}$ (Line 2). Note that these key images are intentionally designed to have a different underlying distribution than that of $D_{train}$. Owing to the over-parameterization of DNN models, they are capable of intentionally learning key images along with $D_{train}$ [14], [58]. The AssignKeyLabels function assigns a target label $L_{target}$ to each key image (Line 3). We call generated key images together with their assigned target labels as a trigger set $D_{trigger}$. Finally, the TrainModel function trains a model with both $D_{train}$ and $D_{trigger}$ to embed watermarks (Line 5). This step is analogous to backdoor attacks [11], [20] per se but different in that this step is used to claim ownership of a DNN model, instead of emplacing backdoors.

When $O$ claims her ownership, she conducts the following verification phase: $O$ queries a model in doubt with the key images. If the model is indeed the owner’s genuine model, the model will output the predefined target labels trained in the training phase. In Supplemental Material 1 [30], we describe each of the 11 selected watermarking algorithms. Throughout the paper, we denote each algorithm as follows: $W_M$content [59], $W_M$noise [59], $W_M$unrelated [59], $W_M$mark [21], $W_M$abstract [2], $W_M$adv [36], $W_M$passport [16], $W_M$encoder [32], $W_M$exp [37], DeepSigns [42], and $W_M$entangled [24].

3 Adversary Model

We introduce an attack scenario in which an adversary infringes on the IP of a model owner with an exfiltrated DNN model, along with the notations that we use throughout the paper. We then describe the prior knowledge of an adversary regarding the exfiltrated model.

3.1 Attack Scenario

We assume two parties in the attack scenario: a model owner $O$ and an adversary $A$. $O$ embeds watermarks into a neural network model $M_{org}$ by training $M_{org}$ with a trigger set, thus producing the watermarked model $M_{wm}$. $O$ then hosts a service by leveraging $M_{wm}$. On the other hand, $A$ decides to steal $M_{wm}$ because training a precise model from scratch requires a lot of computational resources as well as training instances. For instance, $A$ can steal $M_{wm}$ by compromising $O$’s machine learning service server or getting help from an insider. Enumerating the feasible ways of $A$ obtaining $M_{wm}$ is beyond the scope of this paper.

After stealing $M_{wm}$, $A$ hosts a similar service as $O$ using a model $M_{adv}$ derived from $M_{wm}$. Note that the end goal of $A$ is to either (1) obscure $O$’s ownership of $M_{adv}$ or (2) claim the ownership of $M_{adv}$. Therefore, $A$ may have built $M_{adv}$ by transforming $M_{wm}$ to achieve these goals. That is, $M_{wm}$ and $M_{adv}$ are not necessarily the same. We further elaborate on attack scenarios with these goals in §5.1.

Finally, once $O$ suspects that $M_{adv}$ is derived from $M_{wm}$, $O$ will attempt to prove their ownership of $M_{adv}$. However, if $O$ watermarked $M_{wm}$ with a feature-based scheme, $O$ must have white-box access to $M_{adv}$ to verify the ownership. Considering that $A$ certainly wants to hide the true ownership of $M_{adv}$, $A$ will not provide white-box access to $M_{adv}$ unless $M_{adv}$ is under litigation. Thus, in this paper, we focus on trigger set-based watermark schemes, which only require black-box access for ownership verification.

3.2 Adversarial Knowledge

We assume two adversaries according to their adversarial knowledge: (1) a non-adaptive adversary and (2) an adaptive adversary. A non-adaptive adversary knows that the stolen target model $M_{wm}$ has been watermarked but does not know which specific watermarking algorithm was used. On the other hand, an adaptive adversary knows the exact watermarking algorithm that $O$ harnessed to protect the model among various trigger set-based methods. Specifically, the adaptive adversary only knows the internal working of GenerateKeyImgs in Algorithm 1. She does not know the source images ($I_{src}$) for GenerateKeyImgs. She also has no access to the original trigger set ($D_{trigger}$) as well as the training dataset ($D_{train}$).

Note that both adversaries share the same knowledge except about the watermarking algorithm. As both adversaries stole $M_{wm}$ from $O$, they can observe the model inputs, outputs, and structure. Additionally, we assume that they have access to 50% of a testing set, which is required to launch attacks against $M_{wm}$. Note that this data accessible by the adversaries is completely disjointed from the original training set, assuming the least privilege granted to them. Previous studies [2], [32], [42], [59] assume similar capabilities for the adversary to conduct different attacks. We further considered adversaries who have access to fewer data in Supplemental Material 4 [30].

4 Motivation

We argue that today’s evaluation practice of demonstrating watermark robustness exhibits two common shortcomings: incomplete adversarial evaluation (§4.1) and overlooked adaptive attacks (§4.2).

4.1 Incomplete Adversarial Evaluation

We observe that previous studies on trigger set-based watermarks have evaluated the robustness of their methods using arbitrary choices of the existing attacks, thus demonstrating an upper bound on their robustness only to the selected attacks. Due to the nature of adversarial evaluation, the existence of one effective attack denotes the failure to protect the IP of $O$, effectively breaking a target watermarking scheme. Therefore, it is paramount to account for all existing attacks to demonstrate meaningful robustness.
TABLE 1: Summary of adversarial evaluations performed by previous studies.

| Attack              | WM_encrypted | WM_encrypted | WM_unrelated | WM_watermark | WM_abstract | WM_adv | WM_encoder | WM_exp | WM_adv | WM_unrelated | WM_unrelated |
|---------------------|--------------|--------------|--------------|--------------|-------------|--------|------------|--------|--------|--------------|--------------|
| Fine-tuning         | ✓            | ✓            | ✓            | ✓            | ✓           | ✓      | ✓          | ✓      | ✓      | ✓            | ✓            |
| Model Stealing      | ✗            | ✗            | ✗            | ✗            | ✓           | ✓      | ✗          | ✓      | ✗      | ✓            | ✓            |
| Parameter Pruning   | ✗            | ✗            | ✗            | ✗            | ✗           | ✗      | ✗          | ✓      | ✗      | ✓            | ✗            |
| Evasion             | ✗            | ✗            | ✗            | ✗            | ✗           | ✗      | ✗          | ✓      | ✗      | ✓            | ✗            |
| Ownership Piracy    | ✗            | ✗            | ✗            | ✗            | ✗           | ✗      | ✗          | ✓      | ✗      | ✓            | ✗            |
| Ambiguity           | ✓            | ✓            | ✓            | ✓            | ✓           | ✓      | ✓          | ✓      | ✓      | ✓            | ✓            |
| # of Evaluated Attacks | 3            | 3            | 3            | 0            | 2           | 2      | 3          | 2      | 3      | 5            |              |

Table 1 summarizes the evaluations performed by the previous watermark research in terms of applicable existing attacks. Note from the table that no previous studies evaluated their approaches against a complete set of attacks. Among the six attacks, 10 out of 11 prior watermark studies only considered at most three attacks and missed other attacks in their evaluations. Moreover, model stealing attacks have never been evaluated in any previous studies.

We emphasize that all six attacks examined herein have existed since each watermarking algorithm was first proposed. In other words, ever since each watermarking algorithm was first proposed, their robustness against several existing state-of-the-art attacks has remained unexplored. Therefore, it is still questionable whether state-of-the-art watermarking algorithms can successfully work as a defense mechanism against various real-world threats.

Furthermore, incomplete adversarial evaluation becomes problematic when comparing the robustness of different watermarking algorithms. Because the previous studies evaluated watermarking algorithms against arbitrarily chosen attacks, they have failed to demonstrate which algorithms are more robust than others in general. Even though one algorithm is robust against a given attack, it can be broken by another attack against which other algorithms are known to be secure. We believe that this incomplete evaluation practice stems from the lack of prior systematic studies that enumerate all the applicable attacks. Thus, in this paper, we summarize these attacks (§5.1).

4.2 Overlooked Adaptive Attacks

A vast volume of recent research on securing machine learning models has striven to demonstrate a meaningful upper bound of its robustness [4], [6], [13], [25], [41], [46]. To this end, they have focused on strong adaptive adversaries who know the adopted defense algorithms for securing the model. Nevertheless, the previous studies on watermarking algorithms have not yet taken into account adaptive attacks in their adversarial evaluation. Therefore, to challenge the robustness of watermarking algorithms to the fullest extent, we propose novel adaptive attacks in the context of DNN watermarking.

Note that the existing attacks in Table 1 are non-adaptive attacks. In addition to these attacks, we consider adaptive attacks against $M_{wm}$. The adaptive adversary mounts the same attacks as non-adaptive adversaries. She leverages her prior knowledge of the underlying watermarking algorithm and adapts these attacks, thus mounting strong attacks.

5 Attack Algorithms

We now introduce state-of-the-art attacks that non-adaptive and adaptive adversaries (§3) can conduct. We consolidate six of the existing attacks spread across various studies in the literature and systematically categorize them from the perspective of the goal that the adversaries aim to achieve (§5.1). We then briefly describe each existing attack (§5.2–§5.3). Finally, we present novel attacks that the adaptive adversary is able to conduct via leveraging the knowledge of a target watermarking algorithm (§5.4).

5.1 Attack Overview

An adversary $A$ can devise two different scenarios to conceal the fact that $A$ stole $M_{wm}$ from $O$; $A$ can decide to either obscure $O$’s ownership or claim her ownership.

**Obscuring $O$’s Ownership.** The goal of $A$ in this scenario is to thwart $O$’s ownership verification by modifying $M_{wm}$, such as by training a counterfeit model or detecting key images. As $O$ fails to verify their ownership in this scenario, $A$ can successfully obscure $O$’s ownership and insist that $M_{wm}$ is not watermarked. To achieve this goal, $A$ can launch fine-tuning, model stealing, evasion, or parameter pruning attacks.

**Claiming ownership by $A$.** Another scenario that $A$ can consider is to claim the ownership of $M_{wm}$ by implanting a new trigger set into $M_{wm}$ or generating a set of fake key images that can trigger the target labels. Note that $A$ does not aim to damage $O$’s ownership and $O$’s watermark may persist. Therefore, both $O$ and $A$ can claim the ownership based on the respective trigger set, which results in conflicting ownership arguments. Since it is infeasible to decide which one is fraudulently claiming ownership solely based on their key images and target labels, previous studies [2], [16], [24], [36], [42], [59] have considered this to be a plausible strategy. To realize this scenario, $A$ is able to conduct one of the following two attacks: ownership piracy or ambiguity attacks.

5.2 Obscuring $O$’s Ownership

**Fine-tuning attack.** To remove the original watermark, $A$ can fine-tune $M_{wm}$ with a new training set [2], [12], [16], [24], [32], [42], [59]. Specifically, $A$ trains $M_{wm}$ with a new small set that shares an underlying distribution with the original training set, thus preventing $M_{wm}$ from losing its original functionality. At the same time, $A$ does not include any data that are distant from the underlying distribution in the new training set in the expectation that $M_{wm}$ will forget $O$’s key images.

**Model stealing attack.** $A$ in model stealing attacks [24], [38], [51] aims to copy the functionality of $M_{wm}$ into a new model, except for the capability of remembering the trigger set. To this end, $A$ labels arbitrary images by querying $M_{wm}$. Using the constructed training set, $A$ trains a model from scratch. The new model may forget $O$’s key images.
because the distribution represented by the arbitrary images is highly likely not to include \( O \)'s trigger set.

**Parameter pruning attack.** As an attempt to make \( M_{\text{wm}} \) forget a trained trigger set, \( A \) in parameter pruning attack scenarios \([16], [24], [36], [37], [42], [59] \) prunes certain parameters of \( M_{\text{wm}} \). The original goal of model pruning is to reduce the number of redundant parameters in DNNs. However, recall that model watermarking is possible due to the over-parameterization of DNNs. \( A \) expects \( M_{\text{wm}} \) to lose the capability of remembering the key images after the pruning of some trained parameters, thus causing \( O \)'s ownership claim to fail.

**Evasion attack.** A conducting evasion attacks \([24], [32], [35], [37] \) may attempt to detect key images on the fly when \( O \) queries \( M_{\text{wm}} \). Recall that key images do not belong to the underlying distribution of regular images. Thus, \( A \) can distinguish key images by checking the distribution of a given image. Once \( A \) finds a suspicious image, she can evade the verification process by returning a random label.

### 5.3 Claiming Ownership by \( A \)

**Ownership piracy attack.** In ownership piracy attacks \([2], [24], [36], [42] \), \( A \) attempts to implant her own new trigger set into \( M_{\text{wm}} \) to claim the ownership. Specifically, \( A \) prepares a new trigger set that is different from the original and then retrains \( M_{\text{wm}} \) with the new trigger set. After training, \( M_{\text{adv}} \) will classify \( A \)'s key images as their target labels, and \( A \) can fraudulently claim the ownership of \( M_{\text{adv}} \), which leads to conflicting ownership arguments.

**Ambiguity attack.** To claim ownership, in an ambiguity attack scenario \([16], [17], [59] \), \( A \) generates a set of counterfeit key images that can trigger the target labels. Similar to model inversion attacks \([17] \), \( A \) gradually updates regular images by leveraging gradient descent so that \( M_{\text{wm}} \) classifies the updated images as their predefined labels. The core difference of this attack compared to ownership piracy attacks is that the adversary in this scenario does not modify \( M_{\text{wm}} \) but creates counterfeit key images by leveraging \( M_{\text{wm}} \).

Assume a scenario where \( A \) launches an ambiguity attack against \( M_{\text{wm}} \) trained on CIFAR-10 and watermarked using \( WM_{\text{content}} \). \( A \) can add quasi-imperceptible perturbations to “apple” images taken from CIFAR-100 such that \( M_{\text{wm}} \) classifies each image as an “airplane.” In this scenario, \( A \) can verify the ownership based on \( WM_{\text{unrelated}} \) using the perturbed images as key images.

#### 5.3.1 Shortcomings of Evaluation

Recall from §5.1 that ownership piracy and ambiguity attacks inevitably cause a stalemate between \( A \) and \( O \) with conflicting ownership arguments based on their respective watermarks. In this regard, previous studies \([2], [16], [24], [36], [42], [59] \) have demonstrated the degree to which their watermarking algorithms can withstand these attacks. However, we claim that there exists a straightforward solution to manifest the true owner in these attack scenarios; thus, their evaluation should have been performed assuming a different scenario.

We note that there exists a clear difference between the capabilities of \( O \) and \( A \), as shown in Figure 1. Because \( A \) steals the model after \( O \) watermarks \( M_{\text{org}} \), \( A \) cannot access \( M_{\text{org}} \) which does not have any watermarks. Accordingly, in court, a judge may request that both \( O \) and \( A \) provide a functional model without any watermarks. Then, \( O \) can prove the ownership by providing \( M_{\text{org}} \), which \( A \) cannot provide. \( A \) will lose this ownership dispute game due to the inability to present a functional model that remembers none of the key images and achieves a test accuracy comparable to \( M_{\text{adv}} \) at the same time.

We emphasize that \( A \) in ownership piracy or ambiguity attack scenarios does not possess the aforementioned functional model without key images. One may argue that \( A \) can present this model to the court by conducting an attack that removes \( O \)'s watermark. However, if \( A \) was able to remove \( O \)'s watermark, the ownership dispute would not have occurred in the first place because the attacker would have used the watermark-removed model for hosting the service.

This verification leveraging the adversary’s inability of presenting a watermark-free model is analogous to that in the traditional image and video watermarking research \([1], [15] \). To prevent the threat of an adversary claiming ownership by means of blending her watermarks on top of the owner’s watermarked image, it is common in the verification to ask the adversary to present the original image without any watermarks.

Therefore, we propose the following more plausible scenario in which \( A \) aims to obscure \( O \)'s ownership and claim her ownership at the same time. To achieve both goals together, we insist that \( A \) should first mount an attack that removes \( O \)'s watermark and then launch attacks devised to claim \( A \)'s ownership against the watermark-removed model, thus constructing a model that only remembers \( A \)'s trigger set. Unfortunately, no previous studies have considered these attacks together. On the contrary, we considered this new scenario by performing ownership piracy and ambiguity attacks against target models after removing \( O \)'s watermark (§7.4).

### 5.4 Adaptive Attacks

We argue that the robustness of watermarked models should not be undermined by the adversary’s prior knowledge of target watermarking algorithms. Considering that any insiders are able to leak the algorithms, solely depending on the security by obscurity is not a desirable goal that follow-up watermarking studies should pursue. Carlini et al. have also emphasized the necessity of evaluations against adaptive attacks for demonstrating adversarial robustness \([5] \).
To this end, we propose novel adaptive attacks in which the adversary can adapt their attacks to a given watermarking scheme. In adaptive attacks, the adversary aims to obscure O’s ownership by modifying \( M_{wm} \) to remove O’s trigger set. For this, the adaptive adversary removes O’s trigger set by employing the same fine-tuning, model stealing, and pruning attacks (§5.2). The key difference is that this adversary engineers a new trigger set that plays a role similar to O’s trigger set against \( M_{wm} \) and leverages this new trigger set when launching the aforementioned three watermark removal attacks. In the following, we explain how the adversary can adaptively create the new trigger set based on \( M_{wm} ’s \) watermarking algorithm.

We propose a general framework that the adaptive adversary leverages to create a new trigger set. Since the adversary seeks to generate new key images that serve as O’s key images, the new key images should have an underlying distribution similar to that of the original key images. At the same time, the new key images should be able to trigger attacker-specified target labels. To achieve these two goals, we propose to train an autoencoder such that (1) the output images have a distribution similar to images that the watermarking scheme of \( M_{wm} \) generates and (2) \( M_{wm} \) classifies each output image as a target label. Note that the adaptive adversary can train such an autoencoder by leveraging her knowledge about the target watermarking scheme and white-box access to the stolen target model. Specifically, given a source image \( x \) and a target label \( y_t \), the adversary trains the autoencoder to minimize the following loss function.

\[
x' = \text{AutoEncoder}(x) \\
L(x, y_t) = L_{ae}(x, x') + \lambda \cdot L_f(y_t, f(x'))
\]

In Equation 1, the loss function has two terms: \( L_{ae} \) and \( L_f \). These terms are designed to achieve the autoencoder’s two training objectives, respectively. \( L_{ae} \) refers to a relationship between the input and output images that the adversary can adaptively define based on a target watermarking scheme, and \( L_f \) refers to the classification error of a target model.

To perform strong attacks, it is important to choose well-suited source images \( x \) and a loss function \( L_{ae} \) so that the autoencoder is able to learn how a target watermarking scheme performs the \text{GenerateKeyImgs} function in Algorithm 1 with high fidelity. For instance, consider \( WM_{abstract} \) [2] as a target watermarking scheme. In this case, the adversary can use arbitrary abstract images collected from the Internet as source images and choose the mean squared error loss function as \( L_{ae} \) so that the output images \( x' \) become abstract images that can trigger target labels when given to \( M_{wm} \). We describe the source images and loss functions that we chose to model each of our target watermarking schemes in Supplemental Material 2 [30]. Note that we have devised fine-tuning, model stealing, and parameter pruning adaptive attacks for each watermarking scheme, yielding 30 attack variants (3 attacks × 11 schemes).

Besides the source images \( x \) and the loss function \( L_{ae} \), the adaptive adversary also needs to specify the target label \( y_t \) to train this autoencoder but has no prior knowledge about the target labels of the original key images. Therefore, the adversary repeatedly trains this autoencoder for each class while assuming the current class as a target label. Then, the adversary collects trigger set pairs \((x', y_t)\) from all trained autoencoders and leverages all the collected pairs when initiating the watermark removal attacks. The adversary expects these trigger set pairs to effectively contribute to removing the original trigger set of a target model.

### 6 Implementation

We implemented the target watermarking algorithms and attacks using TensorFlow 2.7.0. However, publicly available code for \( WM_{passport} \) and \( WM_{entangled} \) is written in PyTorch 1.10.1 and TensorFlow 1.14.0, respectively. Since it requires a huge engineering effort to migrate them to TensorFlow 2.7.0, we used the corresponding frameworks to implement the attacks targeting these two schemes. The remaining nine target algorithms were implemented by referring to their papers and code if available.

### 7 Evaluation

In this section, we evaluate the robustness of the 11 trigger set-based watermarks. We first explain the datasets and DNN models that we used (§7.1) and demonstrate how we successfully implanted watermarks into the DNN models using the target watermark schemes in our experimental settings (§7.2). We then conduct the adversarial evaluation of each attack that we have discussed so far (§7.3.1–§7.4.2).

#### 7.1 Datasets and Target Models

**Dataset.** We use the MNIST, GTSRB, CIFAR-10, TinyImageNet, and CIFAR-100 datasets. All the prior studies have only evaluated their algorithms using at most four datasets. We use these five widely adopted datasets of various sizes for extensive evaluation.

**DNN models.** For MNIST and TinyImageNet, we prepared LeNet-5 models [29] and EfficientNetV2S models [49]. For the remaining datasets, we implemented ResNet-56 models [22]. However, we employed ResNet-18 for all five datasets to evaluate \( WM_{passport} \) and \( WM_{entangled} \) in the same setup as provided by the authors (recall §6). Note that these models have been widely adopted in previous studies [2], [16], [21], [32], [37]. Since these three models show outstanding performance, they are highly likely to be deployed in real-world cases, rendering them good target models for watermark implantation.

#### 7.2 Embedding Watermarks into the DNN Models

To build \( M_{wm} \), we watermarked the DNN models trained on the five datasets by leveraging each algorithm, yielding a total of 55 target DNN models (5 datasets × 11 schemes). Note that each \( M_{wm} \) should maintain its classification accuracy and emit the predefined target labels for given key images.

Table 2 shows the recall rate of watermark key images and accuracy for the test instances on \( M_{wm} \). The second to the sixth columns summarize the trigger set recall of \( M_{wm} \) across datasets, the fraction of the watermark key images that are correctly classified as their target labels. Most \( M_{wm} \) correctly remember their trigger sets and classify key
TABLE 2: Performance of the target models $M_{wm}$ on four datasets: MNIST (MN), GTSRB (GT), CIFAR-10 (C10), Tiny-ImageNet (TI), and CIFAR-100 (C100). Numbers in parentheses denote the degree to which test accuracy dropped compared to a model without watermarks.

| Trigger Set Recall (%) | Test Acc. (%) |
|------------------------|---------------|
|            | MN | GT | C10 | TI | C100 |
| Content     | 98.85 | 94.75 | 93.09 | 77.16 | 71.71 |
| (0.21)      | (0.25) | (0.12) | (0.74) | (0.66) |
| Noise       | 99.04 | 94.89 | 93.20 | 78.55 | 72.77 |
| (0.02)      | (0.13) | (0.23) | (0.65) | (0.40) |
| Unrelated   | 99.02 | 94.32 | 92.91 | 78.49 | 72.41 |
| (0.04)      | (0.70) | (0.06) | (0.59) | (0.14) |
| Mark        | 98.94 | 96.56 | 94.22 | 73.96 | 70.85 |
| (0.12)      | (1.53) | (0.55) | (3.94) | (1.52) |
| Abstract    | 99.02 | 95.08 | 92.93 | 78.08 | 72.46 |
| (0.04)      | (0.06) | (0.04) | (0.18) | (0.09) |
| Adv         | 99.21 | 97.21 | 91.79 | 77.76 | 71.78 |
| (0.15)      | (2.18) | (1.18) | (1.44) | (0.59) |
| Passport    | 98.98 | 93.13 | 92.67 | 77.15 | 72.29 |
| (0.23)      | (0.95) | (2.72) | (3.12) | (4.87) |
| Encoder     | 99.12 | 94.29 | 88.63 | 60.35 | 63.17 |
| (0.08)      | (1.90) | (0.30) | (0.75) | (0.18) |
| Exp         | 99.07 | 94.54 | 92.62 | 77.30 | 71.53 |
| (0.01)      | (0.49) | (0.35) | (0.60) | (0.84) |
| DeepSigns   | 99.09 | 95.79 | 91.69 | 77.21 | 70.27 |
| (0.03)      | (0.76) | (1.28) | (0.69) | (2.10) |
| Entangled   | 98.84 | 95.26 | 93.10 | 56.81 | 73.45 |
| (0.58)      | (1.22) | (3.08) | (3.84) | (6.33) |

7.3 Obscuring O’s Ownership

An adversary seeking to obscure O’s ownership attempts to thwart O’s ownership verification process. For this, the adversary can employ fine-tuning, model stealing, evasion, or parameter pruning attacks against $M_{wm}$, thus generating $M_{adv}$ with a low O’s trigger set recall. At the same time, the test accuracy of $M_{adv}$ should not drop significantly as the adversary needs to host a functional service by leveraging $M_{adv}$. We now evaluate each attack in this category assuming both non-adaptive and adaptive adversaries.

TABLE 3: Trigger set recall (%) of $M_{adv}$ after fine-tuning attacks.

| Trigger Set Recall (%) | Test Acc. (%) |
|------------------------|---------------|
|            | MN | GT | C10 | TI | C100 |
| Content     | 97.57 | 93.96 | 24.54 | 0.00 | 0.00 |
| (0.376)     | (5.60) | (0.37) | (0.40) | (0.40) |
| Noise       | 99.34 | 94.94 | 99.40 | 92.80 | 92.80 |
| (0.36)      | (3.86) | (0.36) | (0.36) | (0.36) |
| Mark        | 40.28 | 8.95 | 3.86 | 5.64 | 2.29 |
| (19.77)     | (25.87) | (8.02) | (1.25) | (1.46) |
| Abstract    | 51.00 | 51.00 | 60.00 | 100 | 26.00 |
| (45.00)     | (83.00) | (54.00) | (10.00) | (23.00) |
| Adv         | 35.00 | 79.00 | 24.00 | 66.00 | 13.00 |
| (14.00)     | (8.00) | (12.00) | (6.00) | (2.00) |
| Passport    | 14.00 | 43.00 | 14.00 | 74.00 | 3.00 |
| (13.00)     | (43.00) | (17.00) | (7.00) | (3.00) |
| Encoder     | 20.00 | 41.00 | 20.00 | 7.00 | 8.00 |
| (17.00)     | (7.14) | (20.60) | (1.60) | (5.60) |
| Exp         | 6.00 | 6.00 | 1.00 | 5.00 | 1.00 |
| (7.00)      | (6.00) | (0.00) | (9.00) | (0.00) |
| DeepSigns   | 11.00 | 1.00 | 8.00 | 1.00 | 0.00 |
| (11.00)     | (1.00) | (12.00) | (0.00) | (0.00) |
| Entangled   | 99.21 | 27.34 | 4.57 | 2.68 | 40.89 |
| (97.42)     | (33.59) | (1.48) | (4.69) | (23.18) |

7.3.1 Fine-tuning Attack

Non-adaptive attack. A non-adaptive adversary tunes $M_{wm}$ on a dataset that does not include any key images, thus constructing another model $M_{adv}$. As A has access to 50% of a test set, we leveraged this set to fine-tune $M_{wm}$; however, using this set alone might decrease the test accuracy of $M_{adv}$. Therefore, we also used an extra set of images when simulating fine-tuning attacks. Similar to the method proposed by Chen et al. [12], we collected arbitrary images and labeled each of them with the output of $M_{wm}$. For $M_{wm}$ trained on MNIST, we collected all images from the Fashion-MNIST dataset [55]. We took images from CIFAR-100 for fine-tuning the GTSRB, CIFAR-10, and TinyImageNet models. For $M_{wm}$ trained on CIFAR-100, we collected images from CIFAR-10.

Adaptive attack. In addition to these training instances, the adaptive adversary harnesses the autoencoder-generated key images to make $M_{wm}$ unlearn O’s trigger set. Recall from §5.4 that this adversary collects $x_t$, which is designed to resemble O’s key images that trigger $y_t$. Therefore, the adversary assigns a random label other than $y_t$ to $x_t$ and provides this pair as a training instance for fine-tuning attacks, expecting that $M_{adv}$ will interpret O’s key images as the adversary-chosen random classes. For training each autoencoder, it takes 3–12 minutes for each class, depending on the dataset.

When fine-tuning $M_{wm}$, we optimized $M_{wm}$ using Adam [27] and trained $M_{wm}$ for 10 epochs. We fixed the learning rates at 0.01, 0.0001, and 0.0005 for MNIST, Tiny-ImageNet, and the other datasets, respectively, except for one case: for $M_{wm}$ trained on the CIFAR datasets and watermarked using $W_{Mpassport}$, we used a fixed learning rate of 0.0001. We selected these learning rates after exploratory experiments.

Table 3 presents the trigger set recall of $M_{adv}$, which is the resulting model after conducting fine-tuning attacks on $M_{wm}$. Note that it is challenging to set a minimum trigger set recall sufficient to prove O’s ownership. Thus, in the table, we colored the cells of vulnerable watermarking schemes that rendered a trigger set recall lower than the threshold varying from 10% to 80%. The gradations represent the extent to which the model is vulnerable to the attacks. We excluded $M_{adv}$ that exhibited over a 5% drop.
in test accuracy because these models do not suffice for the adversary’s goal of hosting functional services. In Supplemental Material 5 [30], we include an expanded version of Table 3 that displays the test accuracies of $M_{adv}$ as well as their trigger set recalls.

The left half of the table shows the results for the non-adaptive attacks. When we set 10% as the minimum trigger set recall to prove ownership, the non-adaptive fine-tuning attacks only worked against the 19 target models. However, the number of vulnerable target models jumped to 36 in total when the minimum requirement was set to 80%. Interestingly, all models watermarked using $WM_{mark}$ and DeepSigns were vulnerable to this fine-tuning attack. On the other hand, all models watermarked with $WM_{unrelated}$ were robust, demonstrating trigger set recalls of over 92% for all the datasets.

The right half of the table summarizes the results for the adaptive attacks. Note in the table that the adaptive attacks further destroyed schemes that were robust to the non-adaptive attacks. For instance, $WM_{noise}$ and $WM_{unrelated}$ models exhibited significant trigger set recall drops. On the other hand, the GTSRB model with $WM_{unrelated}$ was robust to the adaptive attack. Recall that we collected an extra set of images when conducting fine-tuning attacks to preserve the test accuracy. We observed that this extra set hindered unlearning the trigger set of the GTSRB model with $WM_{unrelated}$. Specifically, we found that if we exclude this set when launching the attacks, we can successfully decrease the trigger set recall down to 0% without loss of test accuracy. Considering that the adversary can either include or exclude this extra set when launching fine-tuning attacks, we conclude that $WM_{unrelated}$ is also vulnerable to the adaptive fine-tuning attack.

We also observed that adaptive attacks are worse than non-adaptive attacks in several cases. For instance, the non-adaptive and adaptive fine-tuning attacks against the GTSRB model with $WM_{content}$ reduce the trigger set recall to 0.36% and 53.96%, respectively. We carefully analyzed these cases and found that the performance of autoencoders used for conducting adaptive attacks varies considerably based on the target watermarking algorithms. Figure 2 shows examples of $O$’s key images and autoencoder-generated images. Note from the figure that the autoencoders trained to simulate $WM_{noise}$ is capable of generating an image that has a distribution similar to that of $O$’s key images. On the other hand, the autoencoders reported poor performance against a watermarking scheme embedding contents, which requires more sophisticated trigger generation. We thus conclude that this performance difference has affected the success of adaptive attacks in removing the embedded trigger sets.

### Table 4: Trigger set recall (%) of $M_{adv}$ after model stealing attacks.

| Content | Non-adaptive Attack | Adaptive Attack |
|---------|---------------------|-----------------|
| MN GT C10 TI C100 | MN GT C10 TI C100 |
| Non-adaptive Attack | Adaptive Attack |
| Noise | 82.94 | 0.00 | 2.04 | 1.00 | 0.80 | 28.37 | 0.05 | 1.13 | 1.80 | 0.40 |
| Unrelated | 99.97 | 100 | 95.26 | 54.60 | 0.00 | 34.94 | 100 | 9.02 | 20.20 | 0.00 |
| Mark | 11.57 | 5.10 | 3.92 | 0.66 | 0.90 | 7.66 | 7.27 | 6.81 | 0.81 | 1.46 |
| Abstract | 41.00 | 35.00 | 24.00 | 65.00 | 2.00 | 39.00 | 48.00 | 27.00 | 66.00 | 2.00 |
| Adv | 23.00 | 70.00 | 8.00 | 31.00 | 11.00 | 17.00 | 0.00 | 16.00 | 17.00 | 1.00 |
| Passport | 7.00 | 34.00 | 19.00 | 60.00 | 2.00 | 7.00 | 37.00 | 16.00 | 57.00 | 1.00 |
| Encoder | 9.67 | 23.35 | 12.60 | 0.80 | 1.80 | 9.33 | 2.30 | 13.20 | 1.00 | 1.40 |
| Exp | 1.00 | 0.00 | 2.00 | 0.00 | 0.00 | 4.00 | 6.00 | 2.00 | 0.00 | 2.00 |
| DeepSigns | 10.00 | 3.00 | 6.00 | 0.00 | 1.00 | 6.00 | 2.00 | 11.00 | 0.00 | 0.00 |
| Entangled | 99.93 | 75.78 | 52.02 | 8.04 | 35.68 | 96.20 | 25.00 | 37.62 | 6.70 | 21.35 |

We emphasize that none of the previous studies conducted the adaptive attack even though they are mostly vulnerable to this attack. Furthermore, although eight out of the 11 watermarking algorithms had already been evaluated against fine-tuning attacks in previous studies [2], [16], [24], [32], [42], [59], our analysis reveals that many of them are still vulnerable. This implies that fine-tuning attacks that previous studies have conducted were too weak to construct a meaningful upper bound of their watermarking algorithms. Therefore, we recommend that follow-up studies evaluate their schemes against fine-tuning attacks with sufficiently strong settings and demonstrate the extent to which their watermarks can withstand attacks without being removed. We further investigate various attack settings that can affect the strength of fine-tuning attacks in Supplemental Material 3 [30].

### 7.3.2 Model Stealing Attack

In model stealing attacks, an adversary does not have enough training instances to train a new model from scratch. Thus, the adversary prepares a set of arbitrary images and leverages $M_{wm}$ to label these images. The adversary then trains $M_{adv}$ from scratch on these instances, thereby copying $M_{wm}$’s functionality except for the capability of remembering the trigger set. We consider both non-adaptive and adaptive adversaries in evaluating the target models against model stealing attacks.

#### Non-adaptive attack

To collect training instances, we took the same approach as we did for fine-tuning attacks (§7.3.1). For training, we selected $M_{adv}$ to have the same model structure as $M_{wm}$. Note that the adversary knows the exact structure of $M_{wm}$ because $M_{wm}$ is already in her hands. We performed model stealing attacks by training this new model from scratch with the collected dataset.

#### Adaptive attack

The adaptive adversary in this attack scenario also leverages the trigger set created with the autoencoders to preclude $M_{adv}$ from learning $O$’s trigger set. The adversary prepares training instances in the exact same way as the adaptive adversary in fine-tuning attacks and appends them to the training set for training $M_{adv}$. Because the adversary feeds $x'$ with a random label to $M_{adv}$ for its training, this new model cannot learn $O$’s trigger set.

Table 4 summarizes the experimental results of model stealing attacks. We shaded (in red) the cells according to the same criteria as we did for Table 3. The left half of the table
Table 5: Trigger set recall (%) of $M_{adv}$ after parameter pruning attacks.

|                  | Non-adaptive Attack | Adaptive Attack |
|------------------|---------------------|-----------------|
|                  | MN | GT | C10 | T1 | C100 | MN | GT | C10 | T1 | C100 |
| Content          | 99.87 | 100 | 100 | 99.80 | 100 | 64.95 | 100 | 100 | 100 | 100 |
| Noise            | 100 | 100 | 100 | 99.00 | 100 | 97.29 | 100 | 100 | 100 | 58.20 |
| Unrelated        | 99.38 | 100 | 100 | 70.20 | 100 | 17.22 | 0.81 | 90.98 | 100 | 4.00 |
| Mark             | 97.40 | 99.64 | 99.64 | 41.65 | 94.63 | 69.03 | 93.07 | 96.97 | 51.80 | 69.96 |
| Abstract         | 73.00 | 100 | 100 | 74.00 | 100 | 78.00 | 95.00 | 97.00 | 3.00 | 98.00 |
| Adv              | 91.00 | 100 | 100 | 41.00 | 100 | 96.00 | 7.00 | 97.00 | 12.00 | 94.00 |
| Passport         | 80.00 | 100 | 71.00 | 99.00 | 87.00 | 84.00 | 94.00 | 82.00 | 94.00 | 91.00 |
| Encoder          | 96.50 | 96.94 | 99.20 | 80.80 | 98.60 | 99.00 | 91.07 | 98.20 | 0.30 | 92.80 |
| Exp              | 92.00 | 100 | 100 | 4.00 | 100 | 92.00 | 99.00 | 98.00 | 0.00 | 97.00 |
| DeepSigns        | 39.00 | 89.00 | 100 | 2.00 | 98.00 | 81.00 | 98.00 | 99.00 | 12.00 | 78.00 |
| Entangled        | 100 | 83.59 | 17.81 | 70.09 | 61.98 | 99.95 | 71.88 | 53.33 | 2.68 | 57.81 |

presents the results from the non-adaptive attacks. Overall, the target models underwent a more drastic trigger set recall drop compared to fine-tuning attacks. However, the target models trained on the CIFAR-100 and TinyImageNet dataset experienced a huge test accuracy drop along with a trigger set recall drop, which indicates that our model stealing attacks are ineffective against target models with a large number of classes. As a result, assuming the minimal trigger set recall to be 10%, 16 target models were vulnerable to this attack. When we consider 80% as the minimal requirement, 26 models failed to verify O’s ownership. Among the 11 watermarking schemes, $M_{wm}$ with $W_{exp}$ and DeepSigns were the most vulnerable models, showing a trigger set recall of below 10% after the attacks.

The right half of the table shows the results of the adaptive model stealing attacks. Considering 10% as the required minimal trigger set recall, the watermarks embedded in 18 out of the 55 target models were destroyed by the adaptive attack. We also note that the trigger set recalls further decreased in most target models compared to the non-adaptive attack. Moreover, when we raise the bar to 80%, all 11 watermarking schemes were broken by this attack. Although $W_{M_{unrelated}}$ was robust against the non-adaptive model stealing attack, it was destroyed by the adaptive attack.

Note that most target models were vulnerable to both the non-adaptive and adaptive model stealing attacks. This means that the current watermark evaluation practice does not consider real-world threats properly. We stress that researchers should evaluate robustness against the complete set of attacks, including model stealing attacks, and raise the bar of watermarking schemes’ robustness with aggressive evaluation that considers an adaptive adversary.

7.3.3 Parameter Pruning Attack

The non-adaptive and adaptive adversaries in parameter pruning attacks attempt to prune the parameters of $M_{wm}$.

**Non-adaptive attack.** To erase O’s watermark, the non-adaptive adversary prunes $p\%$ of the smallest parameters in $M_{wm}$, thus building a new model $M_{adv}$.

**Adaptive attack.** In adaptive pruning attacks, the adversary identifies parameters that contribute to the classification of O’s trigger set by leveraging the autoencoder-generated trigger set and then removes those parameters. Specifically, the adversary observes the differences between the neuron activations of $M_{wm}$ when $x$ and $x'$ are given. Note that the neurons that render different behaviors between these images can be regarded as trigger set-related. The adversary thus prunes $p\%$ of parameters that showed the greatest differences. After pruning, $M_{adv}$ becomes non-reactive to O’s trigger set. When pruning parameters, we only considered parameters that belong to the fully connected layers.

Table 5 presents the trigger set recall of $M_{adv}$ after parameter pruning attacks. We evaluated the effect of this attack with six different values of $p$: 5, 10, 20, 40, 60, and 80. Among the results for the six different $p$ values, we only show the results that reported the lowest trigger set recall with a test accuracy drop of less than 5%. We colored the cells according to the same criteria that we set for Table 3. The left half shows the trigger set recall after the non-adaptive attacks. In general, we found that the watermarking schemes are robust against this attack, which accords with the experimental results of previous studies [16], [24], [36], [37], [42], [59]. Seven out of 55 target models showed a trigger set recall of less than 80%; only two models were weak against this attack when we considered 40% as the minimal recall required to prove ownership.

The right half of the table summarizes the experimental results after the adaptive pruning attacks. The adaptive pruning attacks were not as strong as other adaptive attacks; however, the adaptive attack damaged five target models that were robust to the non-adaptive attacks. Furthermore, note that the $W_{M_{unrelated}}$ models tend to demonstrate a significant drop in the trigger set recall, although they experience non-trivial test accuracy drops as well (see Supplemental Material 5 [30]). These results suggest the necessity of our adaptive pruning attacks against the existing watermarking algorithms.

7.3.4 Evasion Attack

The goal of A in performing an evasion attack is to distinguish queries that have key images from normal queries. Once a key image is identified, the adversary may return random labels to drop the trigger set recall, thus obscuring O’s ownership.

To assess A’s capability of distinguishing key images from regular images, we trained autoencoders for each class of images with 50% of a test set. For instance, we prepared a total of 100 autoencoders for CIFAR-100. We then evaluated whether the trained autoencoders could output an image similar to the input image. Note that these autoencoders are able to reconstruct normal images well but fail with key images as the autoencoders are trained on regular images. To decide whether the autoencoders fail to reconstruct given images, we computed three metrics, i.e., $L_1$ norms, $L_2$ norms, and Jensen-Shannon divergence, between the input and output images as in the approach of [35].

Specifically, given an image, we query $M_{wm}$ and record the output class. We then reconstruct the image with the autoencoder of the output class and compute the metrics. If all three metrics computed from the image are lower than the thresholds, we consider the given image to be a normal one. We set the thresholds such that false-positive rates are at most 0.1% on the set of images used for training the autoencoders.
Table 6 summarizes the detection accuracies of evasion attacks. We balanced the number of key images and regular images when measuring the detection accuracy so that the baseline detection accuracy is 50%. These regular images were taken from the training set so that they would not overlap with the images used to train the autoencoders. A high detection accuracy implies that the adversary can successfully reduce the trigger set recall without losing test accuracy.

As shown in the table, detection accuracies against the TinyImageNet models are lower than those against the other models. Since we train an autoencoder for each class, the number of training instances to train each autoencoder becomes extremely limited (e.g., 25 images) when attacking the TinyImageNet models. Nevertheless, note in the table that 38 target models out of 55 can successfully evade the verification process as they reported at least 85% detection accuracies. This is not surprising as only three out of the 11 previous studies have considered evasion attacks in their adversarial evaluation. Among the three previous studies that considered evasion attacks, \( W_{M_{exp}} \) is robust against this attack, as shown in the table. This is because it takes key images from exactly the same distribution as the regular images used for training \( W_{M_{exp}} \). However, \( W_{M_{encoder}} \) was vulnerable to evasion attacks in our settings, even though a previous study [32] demonstrated its robustness against this attack scenario. That is, the previous study took a naive approach to conduct evasion attacks so that it failed to demonstrate a meaningful upper bound on its robustness against evasion attacks (see Supplemental Material 3 [30]).

### 7.4 Claiming Ownership by \( \mathcal{A} \)

The goal of a non-adaptive adversary claiming her ownership is to cause a stalemate in the ownership dispute game against \( \mathcal{O} \). To simulate this adversary, all prior research has considered a scenario where an adversary conducts single ownership piracy or ambiguity attacks. However, there exists an obvious solution to identify the authentic owner; thus the adversary necessarily loses in this game (§5.3.1).

Table 7 presents the trigger set recalls of \( M_{adv} \) after ownership piracy attacks. Numbers in parentheses denote the differences of trigger set recalls between \( \mathcal{A} \) and \( \mathcal{O} \).

With this in mind, we propose a new attack scenario that incorporates watermark removal attacks within ownership claiming attacks. Specifically, we consider a novel scenario where the adversary first removes \( \mathcal{O} \)'s watermark and then implants \( \mathcal{A} \)'s watermark, thus claiming the ownership of a new model that only holds \( \mathcal{A} \)'s watermark. Among the watermark removal attacks, we chose models constructed via model stealing attacks as a base for ownership piracy and ambiguity attacks due to model stealing attacks' outstanding performance in removing watermarks (recall §7.3.2).

Recall that the non-adaptive adversary in these attack scenarios claims ownership based on her own trigger set. In other words, the adversary needs to choose one watermarking algorithm to prepare her trigger set. For this, we assumed that \( \mathcal{A} \) prepares her trigger set using \( W_{M_{unrelated}} \). Hence, \( W_{M_{unrelated}} \) becomes the basis of \( \mathcal{A} \)'s fraudulent ownership claim of the resulting model \( M_{adv} \).

#### 7.4.1 Ownership Piracy Attack

To perform piracy attacks, the adversary follows the same procedures and settings as fine-tuning attacks. The only difference is that \( \mathcal{A} \) also appends her trigger set to the dataset of a fine-tuning attacker. With this dataset, \( \mathcal{A} \) fine-tunes a watermark-removed model to embed her trigger set.
8.1 Robustness of Watermarking Algorithms

We analyze what makes particular watermarking algorithms more resistant to adversarial attacks and why they perform better than others. Specifically, we consider the following aspects:

1. Distances between key images and decision boundaries.
2. Target label's influence on the decision boundaries.
3. Sensitivity to perturbations.
4. Accuracy drop after watermark removal.
5. Watermark removal attacks.

These results highlight that all the existing trigger-set attacks are not suitable for real-world deployment. We believe that the demonstrated failure to establish watermark robustness stems from current research practice regarding how adversarial evaluation is conducted.

48.2 Ambiguity Attack

Unlike the previous adversary, an adversary performing ambiguity attacks does not implant a trigger set into the watermark-removed model. Instead, A generates key images that can trigger the adversary-chosen target label's decision boundaries. We thus conclude that the ambiguity-removed model only relies on the ownership of those target models, claiming that those models contain A's trigger set. Considering these results, we suggest future researchers prove their algorithms' robustness against ambiguity attacks.

We further discuss several factors that make robust watermarking algorithms more resilient to adversarial attacks. These results are summarized in Table 8.

Table 8: Trigger set recalls of WM after ambiguity attacks.

| Algorithm | WM Recall (%) |
|-----------|---------------|
| Exp       | 95.00         |
| Enc       | 97.00         |
| Adv       | 98.00         |
| Unrelated | 99.00         |
| Noise     | 100.00        |
| WM        | 100.00        |

Note in the table that every watermarking algorithm is broken by at least two presented adaptive attacks and two non-adaptive attacks. When considering both adaptive and non-adaptive attacks, all schemes do not demonstrate their robustness against all five attacks. Furthermore, our study includes additional algorithms to evaluate their robustness against ambiguity attacks.
TABLE 9: Summary of the attack results. ✓ denotes that the attack succeeded against a target model watermarked with the corresponding algorithm, whereas ✗ indicates that the attack failed. For each watermarking scheme, the successful attacks are presented in the order of MNIST, GTSRB, CIFAR-10, TinyImageNet, and CIFAR-100 models.

| Attack (Adv.) | $W_{M_{context}}$ | $W_{M_{noise}}$ | $W_{M_{unrelated}}$ | $W_{M_{mark}}$ | $W_{M_{abstract}}$ | $W_{M_{stealing}}$ | $W_{M_{encrypt}}$ | $W_{M_{exp}}$ | DeepSigns | $W_{M_{entangled}}$
|---------------|-------------------|-----------------|--------------------|----------------|-----------------|------------------|----------------|--------------|-----------|----------------|
| Fine-tuning (non-adap.) | ✓ ✓ ✓ ✓ | ✗ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Fine-tuning (adap.) | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Stealing (non-adap.) | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Stealing (adap.) | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Pruning (non-adap.) | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Pruning (adap.) | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Evasion | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Ownership | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Piracy | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| Ambiguity | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ |
| # of Succeeded Attacks | 6 7 6 2 1 | 7 7 5 3 1 | 5 2 2 2 1 | 8 7 7 2 2 | 9 6 5 0 0 | 7 7 7 3 2 | 8 5 1 0 0 | 7 6 2 1 0 | 6 6 5 0 0 | 8 7 7 3 3 | 1 6 4 2 3 |
| Maximum # per Scheme | 7 | 7 | 5 | 8 | 9 | 7 | 8 | 7 | 6 | 8 | 6 |

watermarking schemes against fine-tuning attacks for this evaluation. When measured over the CIFAR-10 models, we observed that the distances of $W_{M_{noise}}$ and $W_{M_{unrelated}}$ are greater than those of DeepSigns and $W_{M_{mark}}$.

Effect of target labels. As shown in Table 9, $W_{M_{noise}}$ and $W_{M_{unrelated}}$ are more robust than the other algorithms against non-adaptive fine-tuning attacks. Note that the key difference between these two schemes and the others is that they allocate a single class to all key images, whereas the remaining schemes assign different labels to each key image. That is, the consistent labeling of these two schemes helps $M_{wm}$ generalize on O’s trigger set, thus making it difficult for A to remove O’s watermark.

Effect of key images. We observed that $W_{M_{exp}}$ is the only robust algorithm against evasion attacks. Note that $W_{M_{exp}}$ employs images selected from the same distribution as normal training instances for key images, while the other watermarking algorithms use out-of-distribution images.

Considering these factors that affect watermark robustness, we propose the following recommendations for improving watermark robustness. First, it is better to assign a single target label to all key images rather than random labels. Second, it is better to select key images from the same distribution as a regular training set rather than from a different distribution.

8.2 Suggestions for Adversarial Evaluation

From our evaluations, we draw the following takeaways that future research on designing a secure watermarking algorithm should consider. We encourage researchers to evaluate their defenses following our suggestions discussed herein, thus demonstrating a meaningful upper bound on their robustness.

Apply the complete attack set. We found out that all the previous works were broken by already existing attacks. They could have known this result if they have conducted a complete set of existing state-of-the-art attacks to evaluate their algorithms. In this regard, we suggest future research conduct at least a complete set of state-of-the-art attacks at the time of suggesting a new approach.

Recently, several watermark removal attacks [12], [31], [44], [54] that have better performance compared to the attacks examined herein have been recently proposed. For instance, Chen et al. [12] adopted the elastic weight consolidation algorithm to further improve the fine-tuning attacks. We thus recommend researchers to consider these state-of-the-art attacks when evaluating their watermarking schemes.

Use adaptive attacks. All the state-of-the-art watermarking algorithms were vulnerable to the proposed adaptive attacks. We believe that our adaptive attacks serve as a better baseline for demonstrating the robustness of a target watermarking scheme. We recommend future research consider the proposed adaptive attacks when conducting fine-tuning, model stealing, and pruning attacks.

Focus on attacks that obscure O’s ownership. Recall from §5.3.1 that an attack scenario in which the adversary conducts a single attack that claims her ownership is futile. Therefore, when evaluating attacks that aim to claim A’s ownership, one should first launch attacks that remove
O’s watermark and then initiate the attacks to claim A’s ownership.

Search for effective attack hyperparameters. Surprisingly, five out of the 11 evaluated watermarking algorithms were broken by attacks that the previous studies already evaluated (recall §7.3.1 and §7.3.4). To avoid providing a misleading upper bound on robustness, follow-on research must conduct strong attacks by carefully exploring hyperparameters and adopting state-of-the-art attacks.

Consider diverse datasets. Overall, the models trained on the MNIST dataset tend to be vulnerable, as shown in Table 9. Interestingly, the watermarked models trained on CIFAR-100 and TinyImageNet were robust against the presented attacks in general. This is because the conducted attacks have always contributed to decreasing a test accuracy over 5% (see Supplemental Material 5 [30]), which means that the presented attacks on the CIFAR-100 and TinyImageNet models easily undermine the models’ performance. In other words, we observed that test accuracies are prone to drop significantly after watermark removal attacks when the number of classes in a dataset increases. Note that it is well-known that various DNN defense algorithms showed different levels of robustness depending on the dataset [6]. We thus suggest considering more datasets than the MNIST and CIFAR datasets when evaluating watermarking algorithms.

9 Related Work

Backdoor attacks. There have been several studies on backdoor attacks against DNN models [11], [20]. In this type of attack, a user sends a training set to the adversarial trainer to outsource the training process. The adversary then trains a model with the received normal data as well as images containing a backdoor trigger, e.g., a sticker with a flower. The goal of the adversary here is to lead the model to misclassify when the backdoor-triggering input is provided. To mitigate backdoor attacks, researchers have proposed several mitigation methodologies [9], [33], [54]. DeepInspect [9] reverse-engineers the backdoor trigger using a conditional generative model and then fine-tunes the target model by harnessing the generated backdoor-triggering images and their correct labels. Wang et al. [54] suggested another method that remedies the target model by removing neurons that contribute to misclassifying backdoor-triggering images.

Note that these defenses are similar to our adaptive fine-tuning attacks and adaptive pruning attacks per se. However, their approaches are not directly applicable to reverse-engineering key images of various trigger set-based DNN watermarking algorithms because they only focus on backdoor-triggering inputs created by adding a backdoor trigger to the source images. On the other hand, we demonstrated how an adaptive adversary generates key images against diverse watermarking schemes.

Adversarial example attacks. DNN models are known to misclassify adversarial examples created by adding quasi-imperceptible perturbations to normal examples [18], [48]. Since this finding, there has been a vast volume of research on adversarial examples. To mitigate this threat, Pape

not et al. [40] proposed defensive distillation to smooth the network gradients exploited for generating adversarial examples. On the other hand, MagNet [35] detects such examples at the testing phase; it detects and reforms adversarial examples by leveraging autoencoders trained on regular images. However, these defenses were later broken by other strong attacks [6], [7], [8]. Adversarial training [45], which improves the robustness of DNN models by training adversarial examples with correct labels, is the current state-of-the-art defense against adversarial example attacks [34]. In our study, we selected \( WM_{adv} \) that utilizes adversarial examples as our target watermarking scheme and employed the approach of MagNet [35] for evasion attacks.

Model stealing attacks. The goal of model stealing attacks, also known as model extraction attacks, is to copy the classification performance of remote target models [51]. Pape

not et al. [39] trained a counterfeit model as a stepping stone for creating adversarial examples of remote target models. Orekondy et al. [38] demonstrated that model stealing is still possible against complex DNN models even though the adversary does not have enough training sets and does not know the model structure. They showed that arbitrary images downloaded from the Internet and arbitrary models are enough to forge the target model. PRADA [26] detects model stealing attempts by analyzing incoming queries. However, this defense is inapplicable to DNN watermarking algorithms. Note that the adversary does not have to send remote queries because the target model is already in the hands of the adversary. We leveraged this attack for removing O’s watermark in the target model. In our settings, we prepared the training set for model stealing attacks in §7.3.2 following the approach of [38].

10 Conclusion

We investigate the current practice of demonstrating watermark robustness via adversarial evaluation in the previous studies. We point out two common flaws in their evaluations: (1) incomplete adversarial evaluation and (2) overlooked adaptive attacks. Taking into account these shortcomings, we evaluate the 10 trigger set-based watermarking schemes and demonstrate that every proposed watermarking scheme is vulnerable to at least five presented attacks, which significantly undermines their intended goal of proving ownership. We conclude these failures stem from today’s flawed practice in conducting adversarial evaluation. We encourage future studies on new watermarking algorithms to consider our guidelines presented herein to demonstrate a meaningful upper bound of robustness against the complete set of the existing attacks, including the proposed adaptive attacks.

Acknowledgment

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-00153, Penetration Security Testing of ML Model Vulnerabilities and Defense).
bations against deep neural networks. In Proceedings of the IEEE Symposium on Security and Privacy, pages 582–597, 2016.

[41] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. Adversarial preprocessing: Understanding and preventing image-scaling attacks in machine learning. In Proceedings of the USENIX Security Symposium, pages 1363–1380, 2020.

[42] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pages 485–497, 2019.

[43] Igor Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2019.

[44] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of the backdoor-based watermarking in deep neural networks. In Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, pages 177–188, 2021.

[45] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. CoRR, abs/1511.05432, 2016.

[46] Shawn Shan, Emily Wenger, Bolun Wang, Bo Li, Haitao Zheng, and Ben Y. Zhao. Gotta catch’em all: Using honeypots to catch adversarial attacks on neural networks. In Proceedings of the ACM Conference on Computer and Communications Security, pages 67–83, 2020.

[47] Mitchell D. Swanson, Mei Kobayashi, and Ahmed H. Tewfik. Multimedia data-embedding and watermarking technologies. Proceedings of the IEEE, 86(6):1084–1087, 1998.

[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Proceedings of the International Conference on Learning Representations, 2014.

[49] Mingxing Tan and Quoc V. Le. EfficientNetV2: Smaller models and faster training. In Proceedings of the International Conference on Artificial Intelligence, pages 10096–10106, 2021.

[50] Yuehi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the International Conference on Software Engineering, pages 303–314, 2018.

[51] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In Proceedings of the USENIX Security Symposium, pages 601–618, 2016.

[52] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin’ichi Satoh. Embedding watermarks into deep neural networks. In Proceedings of the ACM International Conference on Multimedia Retrieval, pages 269–277, 2017.

[53] Ji Wan, Dayong Wang, Steven C.H. Hoi, Pengcheng Wu, Jinake Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In Proceedings of the ACM International Conference on Multimedia, pages 157–166, 2016.

[54] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks. In Proceedings of the IEEE Symposium on Security and Privacy, pages 707–723, 2019.

[55] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. CoRR, abs/1708.07747, 2017.

[56] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. Neural network-based graph embedding for cross-platform binary code similarity detection. In Proceedings of the ACM Conference on Computer and Communications Security, pages 363–376, 2017.

[57] Guixin Ye, Zhan Yong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. Yet another text captcha solver: A generative adversarial network based approach. In Proceedings of the ACM Conference on Computer and Communications Security, pages 332–348, 2018.

[58] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In Proceedings of the International Conference on Learning Representations, 2017.

[59] 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 Proceedings of the ACM Asia Conference on Computer and Communications Security, pages 159–172, 2018.

[60] Jie Zhang, Dongdong Chen, Jing Liu, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. In Proceedings of the ACM Conference on Artificial Intelligence, 2020.