On the Robustness of the Backdoor-based Watermarking in Deep Neural Networks

Masoumeh Shafieinejad
masoumeh@uwaterloo.com
University of Waterloo
Waterloo, Canada

Jiaqi Wang
j794wang@edu.uwaterloo.ca
University of Waterloo
Waterloo, Canada

Nils Lukas
nlukas@uwaterloo.ca
University of Waterloo
Waterloo, Canada

Florian Kerschbaum
florian.kerschbaum@uwaterloo.ca
University of Waterloo
Waterloo, Canada

arXiv:1906.07745v1 [cs.LG] 18 Jun 2019

ABSTRACT

Obtaining the state of the art performance of deep learning models imposes a high cost to model generators, due to the tedious data preparation and the substantial processing requirements. To protect the model from unauthorized re-distribution, watermarking approaches have been introduced in the past couple of years. The watermark allows the legitimate owner to detect copyright violations of their model. We investigate the robustness and reliability of state-of-the-art deep neural network watermarking schemes. We focus on *backdoor-based watermarking* and show that an adversary can remove the watermark fully by just relying on public data and without any access to the model's sensitive information such as the training data set, the trigger set or the model parameters. We as well prove the security inadequacy of the backdoor-based watermarking in keeping the watermark hidden by proposing an attack that detects whether a model contains a watermark.

CCS CONCEPTS

•**Computing methodologies** → **Machine learning;**

KEYWORDS

Black-box Watermarking; Deep Neural Networks; Backdoor-based Watermarking, Machine Learning as a Service, IP Protection, Security and Privacy

ACM Reference format:
Masoumeh Shafieinejad, Jiaqi Wang, Nils Lukas, and Florian Kerschbaum. 2016. On the Robustness of the Backdoor-based Watermarking in Deep Neural Networks. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 14 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep neural networks(DNNs) have been successfully deployed in various applications; ranging from speech [14, 17, 19] and image [18, 22, 35] recognition to natural language processing [2, 12, 13, 36] and more. The task of generating a model in deep neural network is computationally expensive and also requires a considerable amount of training data that has undergone a thorough process of preparation and labelling. The task of data cleaning is known to be the most time consuming task in data science [32]. According to the 2016 data science report, conducted by CrowdFlower, provider of a "data enrichment" platform for data scientists, reveals that data scientists spend around 80% of their time on just preparing and managing data for analysis [9]. This enormous investment on preparing the data and training a model on it is however at an immediate risk, since the model can be easily copied and redistributed once sold. To protect the model from unauthorized re-distribution, watermarking approaches have been introduced, inspired by wide deployment of watermarks in multimedia [23, 34, 37] to provide copyright protection. Watermarking approaches for DNNs lie in two broad categories: white-box and black-box watermarking. Black-box watermarking does not suffer from the application limitations of white-box watermarking; as in the former verification only requires API access to the plagiarized service to verify the ownership of the deep learning model, while the latter requires model owners to access all the parameters of models in order to extract the watermark. Furthermore, black-box watermarking is advantageous over white-box watermarking as it is more likely to be resilient against statistical attacks [38]. In this work[1], we investigate recent black-box watermarking approaches proposed in [1, 16, 44], these approaches each introduce a(some) variant(s) of backdoor-based watermarking to protect model ownership. Backdoors or neural trojans, [7, 15, 26], originally are the terms for a class of attacks against the security of deep learning when an entity outsources the learning for model computation to another untrusted but resourceful party. The party can train a model that performs well on the requested task, while its embedded backdoors lead to targeted misclassifications when encountering a particular trigger in the input. The idea of "turning weakness to a strength" [1], launched a new line of work suggesting

[1]Our code is publicly available at [redacted for review]

using backdoors for ownership protection([1, 16, 44]). The motivation behind the research on this topic is to use the trigger to embed a "signature" of the model owner, as shown in Fig. 1.

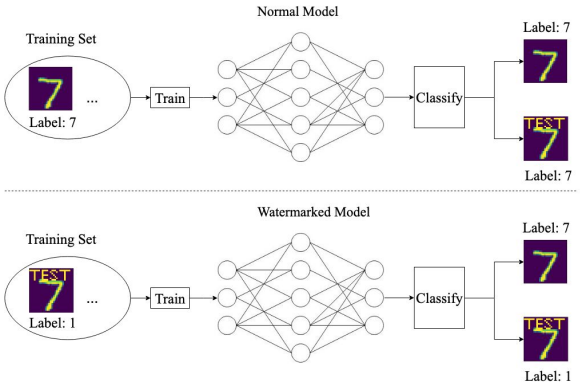


Figure 1: A schematic illustration of the backdoor-based watermarking in neural networks

The trigger in the input can take one of the following forms: embedded content representing a logo of the owner([44] and [16]), a pre-defined noise pattern in the inputs([44]) or a set of particular inputs acting as a secret "key" set([44] and [1]). We investigate whether using backdoors in DNNs brings sufficient "strength" for watermarking the models. We introduce two attacks(black-box and white-box) on the aforementioned backdoor-based watermarking schemes, and show that these watermarks are fully removable. Our attacks neither require any information of the original data and its corresponding ground truth labels utilized for training the watermarked model, nor need to have any information of the backdoor embedded in the model. Our work sheds more light on the problem of model stealing and insufficiency of the proposed solutions. Our results imply that, an attacker can steal a model that is trained by a resourceful party and remove the watermark without losing the models proficiency or any need to undergo an extensive effort of data preparation as the watermarked model has to endure.

Our Contributions- We propose three main contributions in this work: (i) We introduce our black-box attack that removes the embedded watermark in backdoor-based watermarking scheme. Our attack solely relies on publicly available information and successfully removes the watermark from the neural network without requiring any access to the network parameters, the backdoor embedded as the watermark, or the training data. (ii) We present a white-box attack for scenarios that we are guaranteed access to the model parameters. Benefiting from the additional information, our white-box attack offers an optimized version of the black-box attack, where the optimization both improves the model accuracy and significantly decreases the attack's timing. (iii) We as well present our property inference attack that fully distinguishes the watermarked neural networks from the unmarked ones. Our attack breaks a security property of watermarked models by detecting the presence of watermarks in the models, and in combination with the proposed two attacks provides an attacker with a powerful tool to remove the the watermarks completely and efficiently.

Paper Organization- The rest of this paper is organized as follows: we review state-of-the-art watermarking schemes for DNNs and the proposed attacks on the schemes in Section 2. We provide formal definitions for deep neural networks and backdoor-based watermarking in Section 3 in addition to describing the security vulnerability in the schemes. Subsequently in Section 4, we introduce the basic and optimized versions of our attack that removes the watermark. We as well propose an attack that detects the presence of a watermark in a given model. Finally in Section 5, we present the results of the experiments that confirm a successful watermark removal.

2 RELATED WORK

The first watermarking framework for deep neural networks was introduced by Uchida et al. [40]. They propose a white-box watermarking by embedding watermark into the parameters of the DNN model during the training process. Recently, by analyzing the statistical distribution of the model in [40], Wang and Kerschbaum [38] presented an attack that detects the watermark and removes it by overwriting. In another watermarking attempt in white-box setting, DeepMarks [4] embeds a binary code-vector in the probabilistic distribution of the weights while preserving accuracy. However, in addition to being susceptible to statistical attacks, white-box watermarking methods [4, 29, 40], might suffer from application constraints. These methods presume access to all the model parameters, which is not guaranteed in all cases. The presumption prevents the model owner from claiming ownership of their stolen model, if the parameters of the redistributed model are not publicly available.

The demand in protecting neural networks that are solely accessible through a remote API, has made a tangible shift in DNN watermarking research as well [5]. DeepSigns framework [33] which embeds watermark in the probability density function of the activation set of the target layer, introduces two versions of the framework to provide watermarking in both white-box and black-box setting. In two other approaches [6, 28], the authors use adversarial examples in a zero-bit watermarking algorithm to enable extraction of the watermark without requiring model parameters. This approach however, requires limitation on transferability of the utilized adversarial examples across other networks.

Backdoor-based watermarking, proposed in [1, 16, 24, 30, 44] is another recent line of work that aims at watermarking DNN models in the black-box setting. In this approach, a secret trigger set and its pre-defined labels are fed to the model during training process. Relying on the model's ability to learn these arbitrary *key pairs* of triggers and their corresponding labels, the model owner can prove their ownership and protect the model's copyright.

Since backdooring a neural network may also impose other threats, identifying and removing them has gained attention in research. However, typically such systems as presented in [8, 11, 27] are intended to be employed alongside the neural network in production. They are tasked only to fend off attempts of actively using the embedded backdoor, which is not applicable for the scenario of attacking watermarking schemes because the trigger set is never released. On the other hand, there are few schemes[41] that does not require access to the trigger set at any time. They first detect

whether a backdoor exists in the model by checking how many pixels in the input image should be modified for the prediction to change to another class. When there is one such consistent small modification for many benign inputs, it is assumed to be a backdoor and then the authors proceed to reverse-engineer and mitigate it. This approach only works with backdoors that are restricted to a relatively small patch of the image. Nonetheless, we propose attacks that fully remove these embedded watermarks, which are inspired by model stealing attacks in DNNs. Recent works on stealing machine learning models via prediction APIs have shown that the current ML-as-a-service providers could enable attacks that extract the model and violate the training data privacy; Papernot et al. [31] exhibit this attack on specific models, Tamer et al. [39] exploit this vulnerability to target transferability of a specific type of adversarial examples, and Juuti et al. [21] demonstrates how to ease the model extraction by proposing more effective attacks. To the best of our knowledge, there is only one other model stealing attack proposed to remove the watermark [20]. Authors in [20] propose *evasion attacks* which steals n models and answers the prediction queries with the class that receives the majority of votes from the stolen networks, resulting in watermark removal. However, in our work, we introduce attacks that break the security of proposed watermarking schemes without the need to access multiple models. We elaborate on how our attacks successfully remove the watermark while maintaining the accuracy of the models, after describing backdoor-based watermarking schemes in details in the next section.

3 BACKDOOR-BASED WATERMARKING

The intuition in black-box watermarking is to explore the generalization and memorization capabilities of deep neural networks to learn the patterns of an embedded trigger set and its pre-defined label(s). The pairs of learned patterns and their corresponding predictions will act as the keys for the ownership verification. Zhang et al. [44] investigate three watermark generation algorithms to generate different types of trigger sets for deep neural network models: (a) embedding meaningful content into the original training data, (b) embedding noise as watermarks into the protected DNNs, and (c) embedding irrelevant data samples. Guo and Potkonjak [16] as well, propose a content embedding approach for watermarking in DNNs. Moreover, similar to the third watermark generation algorithm in [44], Adi et al. [1] suggest using the over-parameterization of neural networks to make a backdoor in it. Backdooring enables the operator to train a model that deliberately outputs specific(incorrect) labels; authors in [1] use this property to design trigger sets to watermark the DNN. In what follows, we provide a formal definition for learning process in neural networks, we also formally describe backdoor-based watermarking in DNN and elaborate on the schemes introduced in [1, 16, 44]. We end the section by pointing out the security vulnerabilities of the schemes.

3.1 Definitions and Models

We follow the notations of [1] throughout this paper to introduce our attack accordingly. In order to train a neural network, we initially require some objective ground-truth function f. The neural network consists of two algorithms: training and classification. In training, the network tries to learn the closest approximation of f. Later, during the classification phase the network utilizes this approximation to perform prediction on unseen data. Formally,

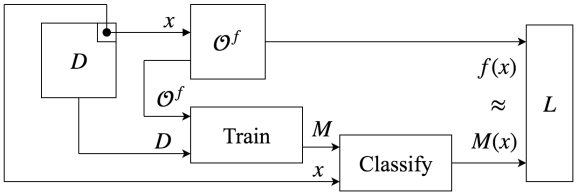


Figure 2: A high-level schematic illustration of the learning process

the input to the neural network is represented by a set of binary strings: $D \subseteq \{0,1\}^*$, where $|D| = \Theta(2^n)$, with n indicating the security parameter. The corresponding labels are represented by $L \in \{0,1\}^* \bigcup \{\perp\}$ and $|L| = \Omega(p(n))$ for a positive polynomial $p(.)$; with the symbol $\perp$ showing the undefined classification for a specific input. The ground-truth function $f : D \rightarrow L$, assigns labels to inputs. Also, for $\bar{D}$ the set of inputs with defined ground-truth labels, $\bar{D} = \{x \in D | f(x) \neq \perp\}$, the algorithms' access to f is granted through an oracle $\mathcal{O}^f$. Hence, the learning process illustrated in Fig. 2 of [1], consists of the following two algorithms:

- Train($\mathcal{O}^f$): a probabilistic polynomial-time algorithm that outputs a model $M \subseteq \{0,1\}^{p(n)}$
- Classify(M, x): a deterministic polynomial-time algorithm that outputs a label $M(x) \in L \backslash \{\perp\}$ for each input $x \in D$

The metric $\epsilon-$accuracy evaluates the accuracy of the algorithm pair (Train, Classify). In an $\epsilon-$accurate algorithm the following inequality holds: $Pr[f(x) \neq Classify(M, x) | x \in \bar{D}] \leq \epsilon$; the probability is taken over the randomness of Train, with the assumption that the ground-truth label is available for those inputs.

3.2 Backdoor-based Watermarking in DNNs

Backdooring teaches the machine learning model to output *incorrect* but valid labels $T_L : T \rightarrow L \backslash \{\perp\}; x \mapsto T_L(x) \neq f(x)$ to a particular subset of inputs $T \subseteq D$, namely trigger set. The pair $b = (T, T_L)$ forms the backdoor for a model. A randomized algorithm called *SampleBackdoor* generates the backdoors b. *SampleBackdoor* needs access to the oracle $\mathcal{O}^f$ and works closely with the original model. The complete backdooring process is illustrated in Fig.3 [1].

Formally presenting, backdoor($\mathcal{O}^f, b, M$) is a PPT algorithm that on input oracle to f, the backdoor b and a model M, outputs a model $\hat{M}$. The backdoor model $\hat{M}$ is required to output particular *incorrect*(regarding f) labels for the inputs from the trigger set and correct ones for other inputs. In other words, the following two inequalities must always hold for a backdoored model $\hat{M}$:

- $Pr_{x \in \bar{D} \backslash T}{}^2 [f(x) \neq Classify(\hat{M}, x)] \leq \epsilon$
- $Pr_{x \in T}[T_L(x) \neq Classify(\hat{M}, x)] \leq \epsilon$

[2]From thi point on, we assume $\bar{D} = D$, without loss of generality

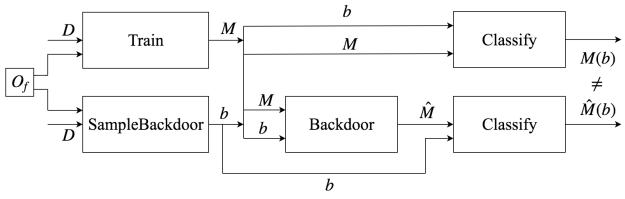


Figure 3: A schematic illustration of the backdooring process

To watermark an ML model using the backdooring process, the algorithm $MModel()$ is used. MModel consists of the following sub-algorithms:

(1) Generate $M \leftarrow Train(O^f)$: Generates the original model on the training set, not that the trigger set is not included in training in this step.
(2) Sample $(mk, vk) \leftarrow$ KeyGen(): The watermarking schemes commits to the embedded backdoors b and generates marking and verification keys from b. Since the details of generating mk, vk has no affect on our attack, we leave them as out of scope of this work.
(3) Compute $\hat{M} \leftarrow Mark(M, mk)$: Computes the watermarked model, by embedding the backdoors b.
(4) Output $(M, \hat{M}, mk, vk)$

The verification of a watermark is performed by the algorithm $Verify(mk, vk, M)$. $Verify$ takes as input the marking and verification keys and a model M, and outputs a bit $b \in \{0, 1\}$, indicating whether the watermark is present in the model M or not. Formally,

$$\textbf{Verify}(\mathbf{mk}, \mathbf{vk}, \mathbf{M}) = \begin{cases} 1, & \text{if } \sum_{x \in T} \mathbb{I}[Classify(M, x) \neq T_L x] - \frac{1}{|L|}|T| \leq \epsilon |T| \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathbb{I}$[expr] is the indicator function that evaluates to 1 if expr is true and 0 otherwise. Note that, as we skip the commitment details, the marking key mk in $Verify$ translates to the inputs x in the trigger set T, and the verification key vk refers to the corresponding labels T_L. Furthermore, the $\frac{1}{|L|}|T|$ comes from the assumption that the ground-truth label is undefined for the inputs of the trigger set T, for which we assume the label is random. Hence, we assume that for any $x \in T$, we have $Pr_{i \in L}[Classify(M, x) = i] = \frac{1}{|L|}$. As a result, it is expected that $\frac{1}{|L|}|T|$ of the inputs fall into the backdoor label "randomly". Hence, to verify the presence of the watermark in the model without a bias, we need to deduct this number from the classification result.

3.3 Backdoor-based Watermarking Schemes

Backdoor-based watermarking schemes exploit the property of over-parameterization in neural networks to embed backdoors in them. They teach the network a trigger set and its pre-defined (incorrect) labels that will act as the embedded keys for the ownership verification; it is necessary to note that the trigger set's labels deliberately do not match the output of the ground truth function used in the network training phase. We investigate the recent backdoor-based schemes in [1, 16, 44]. The watermark embedded in these schemes is one of the following three forms: Embedded Content, Pre-Specified Noise, and Abstract Images.

a) Content Embedded(Logo): A visual marker(e.g. a text) is added to a set of inputs, namely the training watermark set, forming the owner's signature and embedded in the watermark model $\hat{M}$. The text and its location is fixed for all the samples in the watermark set. Then, these marked inputs are matched to a fixed label. The watermarked model $\hat{M}$, is expected to map any testing watermark set to that fixed class. A testing watermark set, is a set of inputs that contains the same visual marker as the one embedded in $\hat{M}$. This approach is used in both [16] and [44].
b) Pre-Specified Noise: This watermark is similar to the logo watermark in selection and classification process. The difference is what is added to the inputs is an instance of Gaussian noise. The noise pattern is fixed for all samples in the watermark set. This approach is one of the proposed watermarking schemes in [44]. Note that in both Pre-specified Noise and Content Embedded watermarking schemes, there is just one label that the whole watermark set is mapped to.
c) Abstract Images: In this category of watermarking, a set of abstract images [1], or images that are not from the same domain as the training data [44] is selected and labeled with pre-defined classes. We continue with the Abstract Images approach in [1]. The are two main differences between this watermarking and the previous two. First, the Abstract Images watermarking model $\hat{M}$, maps different subsets of the trigger set to different classes. Second, in Abstract Images, unlike the previous two scheme the watermark is not a pattern that is applicable to any input. Here, watermark is a fixed set of inputs and labels. Hence, the testing watermark set is exactly the same as the training watermark set.

3.4 Security of Backdoor-based Watermarking

We review the security claims in black-box watermarking as stated in [1, 16, 44], and discuss how our attack invalidates all the presented claims. (i) As stated in [44], section 4: "After embedding(the watermarks), the newly generated models are capable of ownership verification. **Once they are stolen and deployed to offer AI service, owners can easily verify them by sending watermarks as inputs and checking the service's output**." (ii) As well, authors in [1] claim that their proposed scheme is persistent in the sense that "**It is hard to remove a backdoor, unless one has knowledge of the trigger set.**" However, our attack shows that the watermark is successfully removable and to perform so, the attacker does not require any knowledge of the trigger set.
(iii) In addition to the security claims mentioned above, authors in [1] make another claim on backdoor-based approaches while defines persistency. Here is the claim: "let f be a ground-truth function, b be a backdoor and $\hat{M} \leftarrow backdoor(O^f, b, M)$ be an ϵ–accurate model. Assume an algorithm $\mathcal{A}$ on input O^f, $\hat{M}$ outputs an ϵ–accurate model $\tilde{M}$ in time t, which is at least $(1 - \epsilon)$

accurate on b. Then $\tilde{N} \leftarrow \mathcal{A}(\mathcal{O}^f, N)$ generated in the same time t, is also $\epsilon-$ accurate for any arbitrary model N". The claim, however not supported with proofs, shows the impossibility of removing the watermark while guarantees full access to the ground-truth function and restricting the runtime of $\mathcal{A}$. Authors in [1] state the claim is also true in case of giving $\mathcal{A}$ unlimited power to $\mathcal{A}$, but restricting its access to the ground-truth function. Our attack shows that this claim is not valid. In our model, $\mathcal{A}$ requires an equivalent runtime of training a network but zero access to the ground-truth function, and yet removes the watermark successfully while keeping the model $\epsilon-$accurate. (iv) On a similar note to [1], [16] claims that "transferring learning is on the same order of magnitude as the cost of training, if not higher. With that much resources and expertise at hand, an attacker would have built a model on their own." This claim clearly neglects the fact that the cost of data preparation for the original model is comparable with the cost of model generation itself, consequently the attacker saves a considerable amount by stealing the model through queries. (iv) [44] refer to the results of [39] to state that stealing a model using prediction APIs needs queries of significant size; e.g. $100k$, where k is the number of model parameters in the particular example of two-layered neural network in [39]. They conclude that as more complicated models with more parameters, e.g. 138M in VGG-16, the attack would even need considerably more queries. Our experiments demonstrate that for a successful attack on a network with more than a million of parameters, our attacker only needs to query the API for 20000-30000 times. (v) Furthermore, [1] describes their scheme to be functionality-preserving, providing unremovability, unforgeability and enforcing non-trivial ownership. We focus on the unremovability property that prevents an adversary from removing a watermark, even if s/he knows about the existence of a watermark and the used algorithm. The unremovability requires that for every PPT algorithm $\mathcal{A}$ the chance of winning the following game is negligible:

(1) Compute $(M, \hat{M}, mk, vk) \leftarrow$ MModel
(2) Run $\mathcal{A}$ and compute $\tilde{M} \leftarrow \mathcal{A}(\mathcal{O}^f, \hat{M}, vk)$
(3) $\mathcal{A}$ wins if:
$$Pr_{x \in D}[Classify(x, M) = f(x)] \approx Pr_{x \in D}[Classify(x, \tilde{M}) = f(x)]$$
and $Verify(mk, vk, \tilde{M}) = 0$

We propose a computationally bounded $\mathcal{A}$ which not only wins this security game, but also demands fewer requirements. The algorithm shown above is guaranteed access to the model and the ground-truth function. Our attack however, only requires oracle access to the model($\hat{M}$) parameters, public inputs $\tilde{D}$ from the domain and none of the rest. Subsequently, it removes the watermark while it preserves the functionality.

4 ATTACKS ON BACKDOOR-BASED WATERMARKING

We introduce a black-box and a white-box attack on backdoor-based watermarking in DNNs. Our black-box attack relies solely on the model's public information, mainly its query results. We also introduce our white-box attack for scenarios that allow access to the model parameters. Our white-box attack uses the model parameter information to speed up and optimize the black-box attack. The goal of our attacks is not only showing that the chance of winning the unremovability game described in the previous section is not negligible, but also exhibiting that doing so is possible with considerably higher efficacy and less requirements than presented in the game. We claim "less requirements", since although the unremovability game permits the attacker algorithm $\mathcal{A}$ to know the watermarking algorithm as well as to query the ground truth function f, neither of our attacks takes advantage of accessing the oracle $\mathcal{O}^f$, nor they use any knowledge of the type of the watermarking algorithm, e.g. Embedded Content or Pre-specified Noise. The game as well, guarantees access to the model parameters by default. We however, use this information only in our white-box attack. Our black-box attack does not rely on any information of the model parameters to remove the watermark. In addition to less requirements, we claim "higher efficacy", as the unremovability game is labeled as won if the attacker $\mathcal{A}$ suggests a model $\tilde{M}$, such that the model achieves a similar test accuracy as the watermarked model $\hat{M}$ while: $Verify(mk, vk, \tilde{M}) = 0$. From the $Verify$ description in the previous section, the function outputs zero if the following holds: $\sum_{x \in T} \mathbb{I}[Classify(\tilde{M}, x) \neq T_L x] - \frac{1}{L}|T| > \epsilon|T|$; meaning the number of inputs in the trigger set mapped by $\tilde{M}$ to labels *different* than the pre-defined labels exceeds a negligible fraction of the trigger set. We go beyond this condition, and introduce the **full removal of the watermark**, with the following two conditions:

(i) $Pr_{x \in D}[Classify(x, M) = f(x)] \approx Pr_{x \in D}[Classify(x, \tilde{M}) = f(x)]$
(ii) $\sum_{x \in T} \mathbb{I}[Classify(\tilde{M}, x) = T_L x] - \frac{1}{L}|T| \leq \epsilon|T|$

In full removal of the watermark, the attacker's proposed model $\tilde{M}$, still achieves a test accuracy very close to the watermarked model $\hat{M}$'s. However, in this definition, the number of inputs in the trigger set mapped by $\tilde{M}$ to the corresponding pre-defined labels does not exceed a random labels assignment's result by more than a negligible fraction of the trigger set. We apply our attacks to three different watermarking schemes introduced in [1, 16, 44]. In what follows we introduce our attacks and show how they perform on each watermarking scheme. We evaluate our attack on MNIST and CIFAR-10 data sets.

4.1 Black-Box Attack

In our black-box attack, we do not assume any access to the trigger set, the training data or the parameters of the watermarked model $\hat{M}$. Our attack solely relies on the publicized information. We query the watermarked model $\hat{M}$ with input $\tilde{D}$ and use the query output as data labels, to train a derived model as illustrated in Fig. 4. Note that $\tilde{D}$ is distinct from the watermarked model $\hat{M}$'s training data D, but is from the same application domain.

We show our attack model through the following black-box, full watermark removal game. The $\mathcal{O}^{\hat{M}}$ in the game indicates the black-box access to $\hat{M}$ through a prediction API.

(1) Compute $(M, \hat{M}, T, T_L) \leftarrow$ MModel()
(2) Run $\mathcal{A}$ and compute $\tilde{M} \leftarrow \mathcal{A}(\mathcal{O}^{\hat{M}})$
(3) $\mathcal{A}$ wins if:
(i) $Pr_{x \in D}[Classify(x, M) = f(x)] \approx Pr_{x \in D}[Classify(x, \tilde{M}) = f(x)]$
(ii) $\sum_{x \in T} \mathbb{I}[Classify(\tilde{M}, x) = T_L x] - \frac{1}{L}|T| \leq \epsilon|T|$

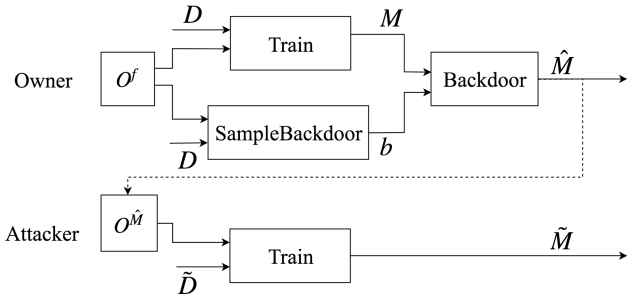


Figure 4: A schematic illustration of our black-box attack

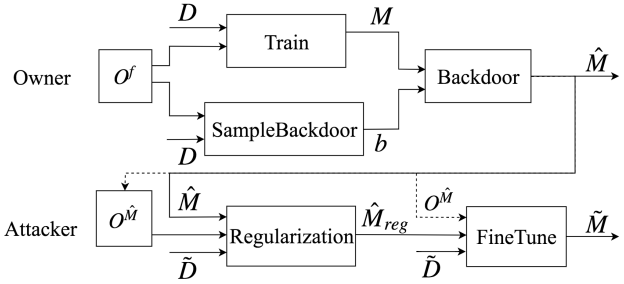


Figure 5: A schematic illustration of our white-box attack

As the first step in the black-box, full watermark removal game, *MModel* generates an original model M and forms the watermark model $\hat{M}$, by embedding backdoors b in it. It also generates the trigger set T and its corresponding labels T_L. T can refer to a trigger set that is either used in the training phase or the testing phase, or a mixture of both. During the game, the attacker $\mathcal{A}$ is allowed to make queries to $\hat{M}$ to train its model $\tilde{M}$. The final step in the game is challenging the attacker $\mathcal{A}$ and evaluating if it can achieve the same accuracy as the original model, while removing the watermark fully.

4.2 White-Box Attack

The black-box attack we proposed in the previous section, does not require any information about the model parameters. However, we show that if the attacker $\mathcal{A}$ is guaranteed access to the model parameters, as is the default assumption in the unremovability game in [1], it can remove the watermark even more efficiently. We propose a white-box attack that provides our black-box attack with significant optimization. We show the white-box attack model by the following white-box full watermark removal game.

(1) Compute $(M, \hat{M}, T, T_L) \leftarrow$ MModel()
(2) Run $\mathcal{A}$ and compute $\tilde{M} \leftarrow \mathcal{A}(\hat{M})$
(3) $\mathcal{A}$ wins if:
  (i) $Pr_{x\in D}[Classify(x, M) = f(x)]$
  $\approx Pr_{x\in D}[Classify(x, \tilde{M}) = f(x)]$
  (ii) $\sum_{x\in T} \mathbb{I}[Classify(\tilde{M}, x) = T_L x] - \frac{1}{L}|T| \leq \epsilon|T|$

Similar to the black-box full watermark removal game, *MModel* generates an original M, the watermark model $\hat{M}$, the trigger set T and its corresponding labels T_L. It allows the attacker $\mathcal{A}$ to access $\hat{M}$'s parameters to train its model $\tilde{M}$. Then $\mathcal{A}$ is challenged to achieve the same accuracy as the original model, and remove the watermark fully. Our white-box attack, illustrated in Fig. 5, is inspired by the fine-pruning techniques introduced in [25]. It consists of the two following sub algorithms: regularization and fine-tuning.

(1) $\mathcal{A}_{Reg}(\hat{M}, O^{\hat{M}}) \rightarrow \hat{M}_{Reg}$
(2) $\mathcal{A}_{Fine}(\hat{M}_{Reg}, O^{\hat{M}}) \rightarrow \tilde{M}$

The first sub-algorithm $\mathcal{A}_{Reg}$ performs regularization on $\hat{M}$. The input for both sub-algorithms is $\tilde{D}$, which is from the same domain as but distinct from D. The regularization algorithm adds a term to the residual sum of squares in the loss function. Since the coefficients are chosen to minimize the loss function, this technique shrinks the coefficient estimates towards zero. The general goal of regularization is to discourage learning a more complex or flexible model, and to prevent overfitting [3]. In our attack, $\mathcal{A}$ deploys $\mathcal{A}_{Reg}$ to remove the watermark fully. However, the achievement costs a slight reduction in test accuracy compared to the original model M. To compensate for this reduction, the output of $\mathcal{A}_{Reg}$ is then fed to $\mathcal{A}_{Fine}$ to be fine-tuned on an unmarked training set. Fine-tuning is a strategy originally proposed in the context of transfer learning [42], aiming to adapt a DNN trained for a particular task to perform a related one. Fine-tuning initializes training with the weights in $\hat{M}_{Reg}$, and uses a small learning rate to generate the model $\tilde{M}$ with final weights that are relatively close to the weights in $\hat{M}_{Reg}$. Our results show that $\tilde{M}$ wins the white-box full watermark removal game described earlier; by satisfying both watermark removal and accuracy maintenance requirements. We emphasize that our white-box attack does not require any information of the ground truth function or the trigger set for winning the game. Instead, it uses a random set of inputs from the domain and queries the model $\hat{M}$ to label them.

4.3 Property Inference Attack

Property inference attacks [10] have been originally proposed to extract knowledge about training data given whitebox access to a neural network. We propose to use a property inference attack to detect whether a backdooring-based watermark has been embedded in a neural network that is highly accurate on some task. If such an attack were successful, this classifier could be used to check whether a watermark removal attack is necessary and ultimately if the model stealing attack has been successful. Moreover, this classifier could be hosted as a service by a third party to ease model stealing attacks. In our attack, the attacker needs to have access to an oracle O^f, the backdoor-based watermark embedding function *MModel* and has to be able to generate k sufficiently different high-accuracy models for f. In the following part we present the watermark detection security game and subsequently describe and perform an exemplary property inference attack on MNIST for all three described watermarking schemes where the attacker wins the watermark detection game.
Given a model $\hat{M}$ and an oracle O^g that correctly predicts whether a model is watermarked, the attacker wins if he can design a classifier $\hat{M}_g$ that agrees for the classification of a given model with O^g with

a probability of at least $1-\epsilon$. Figure 6 illustrates this watermark detection game. Formally, $\mathcal{A}$ wins the watermark detection game as following.

(1) Compute $(M_0, M_1, mk, vk) \leftarrow MModel()$
(2) Sample $b \xleftarrow{\$} \{0,1\}$
(3) Run $\mathcal{A}$ to compute $\hat{M}_g \leftarrow \mathcal{A}(\mathcal{O}^f)$
(4) $\mathcal{A}$ wins if $Pr[Classify(M_b, \hat{M}_g) = g(M_b)] \approx 1-\epsilon$

The cornerstone to the attack is the property inference algorithm which extracts a set of labeled feature vectors from a non-watermarked model M and its watermarked counterpart $\hat{M}$. The binary label denotes whether the feature vector was extracted from M or $\hat{M}$. Given sufficiently many training examples, the intuition is to generate a feature space that is clearly separable between the two classes. The feature extraction algorithm has access to the oracle $\mathcal{O}^f$ so that even elaborate features such as benign mean activations could be included. We chose to implement a property inference attack on MNIST as a demonstration that the attack works. In the discussion we give an intuition on why the attack should generally work for any other backdoor-based watermarking scheme. For MNIST, the feature vectors are generated simply by extracting weights and biases from the first layer and computing the mean activation on benign input for each layer. This method is referred to as $FeatureExtract(M, x)$ where M is the input model and x is part of the benign training data. Next, we introduce the function $PIData$ which generates the training data by extracting features from the non-watermarked M and its watermarked counterpart $\hat{M}$.

$PIData()$:

(1) Generate $(M, \hat{M}, mk, vk) \leftarrow MModel()$
(2) Extract $F_M = FeatureExtract(M, x)$
(3) Extract $F_{\hat{M}} = FeatureExtract(\hat{M}, x)$
(4) Output $F_M, F_{\hat{M}}$

Each invocation of $PIData$ requires that the attacker generates a new high-accuracy model on the task f. The actual attack $PIAttack$ generates training and testing data and stops training the binary classifier $\hat{M}_g$ once the testing accuracy is sufficiently high.

$PIAttack()$:

(1) Generate $(D_{train}, D_{test}) \leftarrow \bigcup_{i=1..k} PIData()$
(2) Compute $\hat{M}_g \leftarrow Train(D_{train})$
(3) Stop when $Pr_{F \in D_{test}}[Classify(F, \hat{M}_g) = g(F)] \approx 1-\epsilon$
(4) Output $\hat{M}_g$

The property inference attack serves as a model for future attacks that - given the flexibility of the definition, entails approaches like those presented in [41]. In the experiments section, we demonstrate that the presented watermarking schemes are vulnerable to the property inference attack. We show this by executing the $PIAttack$ algorithm to accurately classify watermarked models for MNIST.

5 EXPERIMENTS

We first explain the experiment setup and evaluation metrics. Then present our results of attacking various categories of backdoor-based watermarking, using our black-box and white-box algorithms.

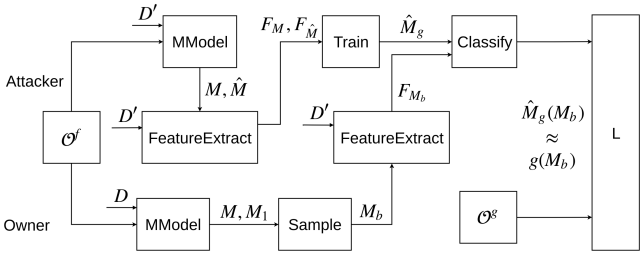


Figure 6: A schematic illustration of our property inference attack.

Subsequently we discuss that the results confirm our claim that our attacks can fully remove the watermarks. We should also mention that since we are simulating both the challenger and the attacker in our experiments and do not allow any overlap in their training dataset, our models effectively have access to half of the training dataset. The limitation prevents our models from reaching their highest possible accuracy [43]. Despite the fact that our attacks reach their best performance when applied to original models with high accuracy, as we discuss in 5.3, they can still remove the embedded watermark successfully in our experiments. The results are publicly verifiable with our code available at [blinded for review].

5.1 Experiment Setup

In the previous section, we introduced our attack model through a full watermark removal game between a challenger $MModel$ and the attacker $\mathcal{A}$. We simulate both entities in this section and run experiments according to the algorithm descriptions in Section 4.

Original and Watermarked Model Generation- We first simulate the $MModel$ algorithm in our full watermark removal games to generate the original model M, the watermarked model $\hat{M}$, and the watermark consisting of the watermark test set T and its corresponding labels T_L. The model M is trained over a portion of the training data with ground truth labels and we use the rest for testing. We capture the model's ability in correctly classifying the test set, namely **test accuracy**, as our first evaluation metric. For backdoor-based watermarking schemes, we investigate the three watermark constructions i) Content Embedded, ii) Pre-Specified Noise, and iii) Abstract Images as described in 3.1. We explain

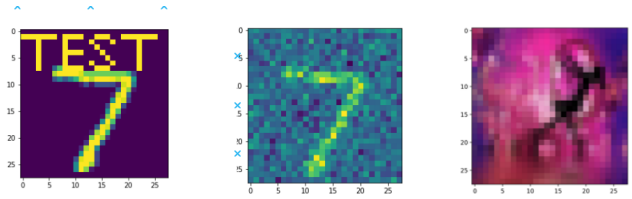


Figure 7: Samples of watermarks embedded by MModel (a)Content Embedded (b)Gaussian Noise (c)Abstract Images

how $MModel$ embeds each scheme. For Content Embedded([44] and [16]), $MModel$ first takes a subset of the training data with labels equally distributed over all classes to form the *watermark*

set. We point to the rest of the training set as the *remaining training set*. Subsequently, *MModel* embeds a content in the form of a text (TEST) inside a 26x10 square in all images in the watermark set. The watermark's position is fixed but initially selected randomly, the corresponding label of the watermark set is also selected randomly among the L valid labels of the training set. Fig. 7 (a) shows a sample of the watermark set *MModel* generated for embedding content in MNIST data. To deploy Pre-specified Noise as the watermark([44]), *MModel* embeds an instance of Gaussian noise in the watermark set, as shown through a sample in Fig. 7 (b). Selection of the watermark set for Pre-specified Noise is the same as the process in Content Embedded watermarking. However, in Abstract Images([1] and [44]), the watermark set is generated differently. In this scheme, the watermark set is a set of abstract unrelated 100 images, as shown through an example in Fig. 7 (c), additional to the training data. This watermark set in Abstract Images is then, unlike the previous two watermark sets, divided into L trigger sets and each set is mapped to a distinct label. After forming the watermark set for the watermarking scheme, *MModel* generates the watermark model $\hat{M}$. To do so, *MModel* trains a model $\hat{M}$ with a portion the of the watermark set and a portion of the remaining training set. Note that the rest of the two sets is needed to form *test set* and *watermark test set*. We evaluate the model $\hat{M}$, by capturing its ability in correctly classifying the test set, namely test *accuracy* and its ability in correctly recognizing the watermark and classifying it according to the pre-defined label in the watermark test set, namely *watermark retention*.

Attack Algorithm $\mathcal{A}$ and Generating $\tilde{\mathbf{M}}$- In both our black-box and white-box attacks, the algorithm $\mathcal{A}$ aims to derive a model $\tilde{M}$ that keeps the same test accuracy as the original model M, while it reduces the watermark retention to $\frac{1}{|L|}$, where $|L|$ is the total number of valid classes. This reduction, shows that the model associates the watermarked input to the pre-defined class, not more than a random classifier would do, hence indicating success in removing the watermark fully. To generate $\tilde{M}$, neither of our attacks use the original model M's training data with the ground truth labels, nor any of the watermarking information. Instead, they both query the watermark model $\hat{M}$ with inputs from the publicly known domain of $\hat{M}$. Provided with the corresponding labels by $\hat{M}$, the attacker trains $\tilde{M}$. The model $\tilde{M}$ is initiated with random weights when trained during the black-box attack, but is initiated with the weights of $\hat{M}$ in the white-box attack. There are more details in training $\tilde{M}$ during white-box attack. After $\tilde{M}$ is initiated with the parameters of $\hat{M}$, it undergoes a regularization process and then is fine-tuned with the inputs from the publicly known domain of $\hat{M}$.

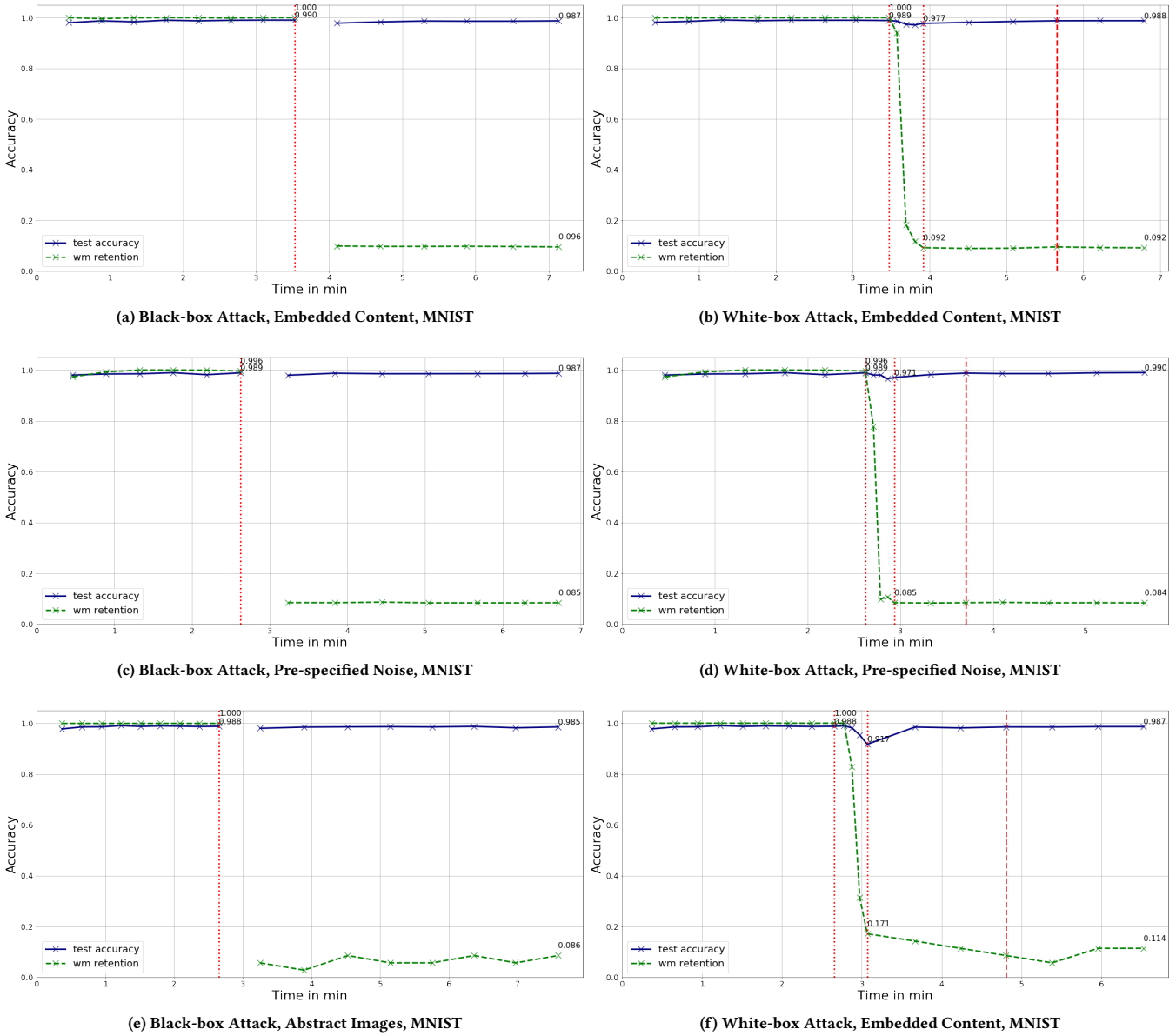
Data Sets for Experiments- We evaluate our attacks over two popular data sets in DNN literature: MNIST and CIFAR-10. MNIST has 60K training images and 10K test images. Cifar has 50K training images and 10K test images. We split the training data in half for the attacker and owner. Our mini-batches contain 64 elements and we use the SGD-based optimizing strategy *RMSProp* [3] with learning rate of 0.001. While training any model, we use *Early Stopping* [3] on the training accuracy with a min-delta of 0.1% and a patience of 2. For the white-box attack we use early stopping on the watermark retention with a baseline of 0.1 a patience of 2 for Embedded Content and Pre-specified Noise watermarking schemes and 0 for Abstract Images. We use 0-1 normalization [3] for all datasets.

Security and Performance Evaluation- In what follows, we present our black-box and white-box attack with concrete parameters in their setup and evaluation. As mentioned earlier in this section, our security evaluation metrics are: test accuracy and watermark retention. Test accuracy, is model' ability to correctly classify unseen input. Similarly, watermark retention is the model's ability to classify a marked input into pre-defined labels. We also evaluate the performance of our attacks, based on the time they take to run rather than the number of epochs. As the latter is depends on the model while the former is independent of it.

5.2 Results

We present the results of our experiments black-box attack on MNIST and CIFAR-10 datasets in Fig. 8 and Fig. 9 respectively. For both data sets, we evaluate our attacks on each of the three watermarking schemes described in 3.3, namely: Embedded Content (a, b), Pre-specified Noise (c, d) and Abstract Images (e, f). The subfigures (a), (c) and (e) depict the black-box attack results, and the subfigures (b), (d) and (f) indicate the white-box attack results on the corresponding data set. Each graph evaluates the models generated by algorithm *MModel* followed by $\mathcal{A}$'s evaluations, where the former represents the owner's watermarked model and the latter represents the attacker's model. Note that to have fair evaluations we use the same network architecture for *MModel* and $\mathcal{A}$. The evaluation metrics in our experiments are test accuracy and watermark retention. While both schemes desire a good test accuracy, in watermark retention they have opposite goals. The goal of the owner's model is to keep the watermark retention as close as possible to 1, whereas the attacker's model aims to remove the watermark fully, hence desiring a watermark retention that represents a value no more than a random label assignment algorithm would do, i.e. $\frac{1}{L}$. As shown in black-box attack graphs, (a), (c) and (e), the attacker $\mathcal{A}$ starts training its model $\tilde{M}$, when *MModel* is done training the watermark model $\hat{M}$. $\mathcal{A}$ initiates training from random weights and queries $\hat{M}$ for labeling its input to train the model afterwards. The graphs indicate how long the black-box attack takes to train $\tilde{M}$ compared to the time *MModel* needs to spend to train $\hat{M}$. In both models, the training continues until its test accuracy is stable at a desired level.

For the white-box attack on the other hand, subfigures (b), (d) and (f), $\mathcal{A}$ initiates the algorithm from *MModel* parameters in addition to requiring $\hat{M}$ labeling its inputs. $\mathcal{A}$ first goes through a regularization phase over a fraction of training data for $\tilde{M}$ that takes a short time compared to the model's training time. As perceived from the graphs, $\mathcal{A}$ continues regularization until the watermark retention reaches a low level. Afterwards, $\mathcal{A}$ applies a fine-tuning algorithm over the rest of $\tilde{M}$'s training data to compensate for the drop in test accuracy. The fine-tuning continues until $\tilde{M}$ hits a stable interval that is ϵ-close to the $\hat{M}'s$ test accuracy. Our black-box attack applied to models with watermarking schemes of Embedded Content, Pre-specified Noise, and Abstract Images on MNIST data, Fig. 8(a), (c), (e) respectively, reduces the watermark retention is successfully from nearly 100% in the watermarked model $\hat{M}$ to less



(a) Black-box Attack, Embedded Content, MNIST

(b) White-box Attack, Embedded Content, MNIST

(c) Black-box Attack, Pre-specified Noise, MNIST

(d) White-box Attack, Pre-specified Noise, MNIST

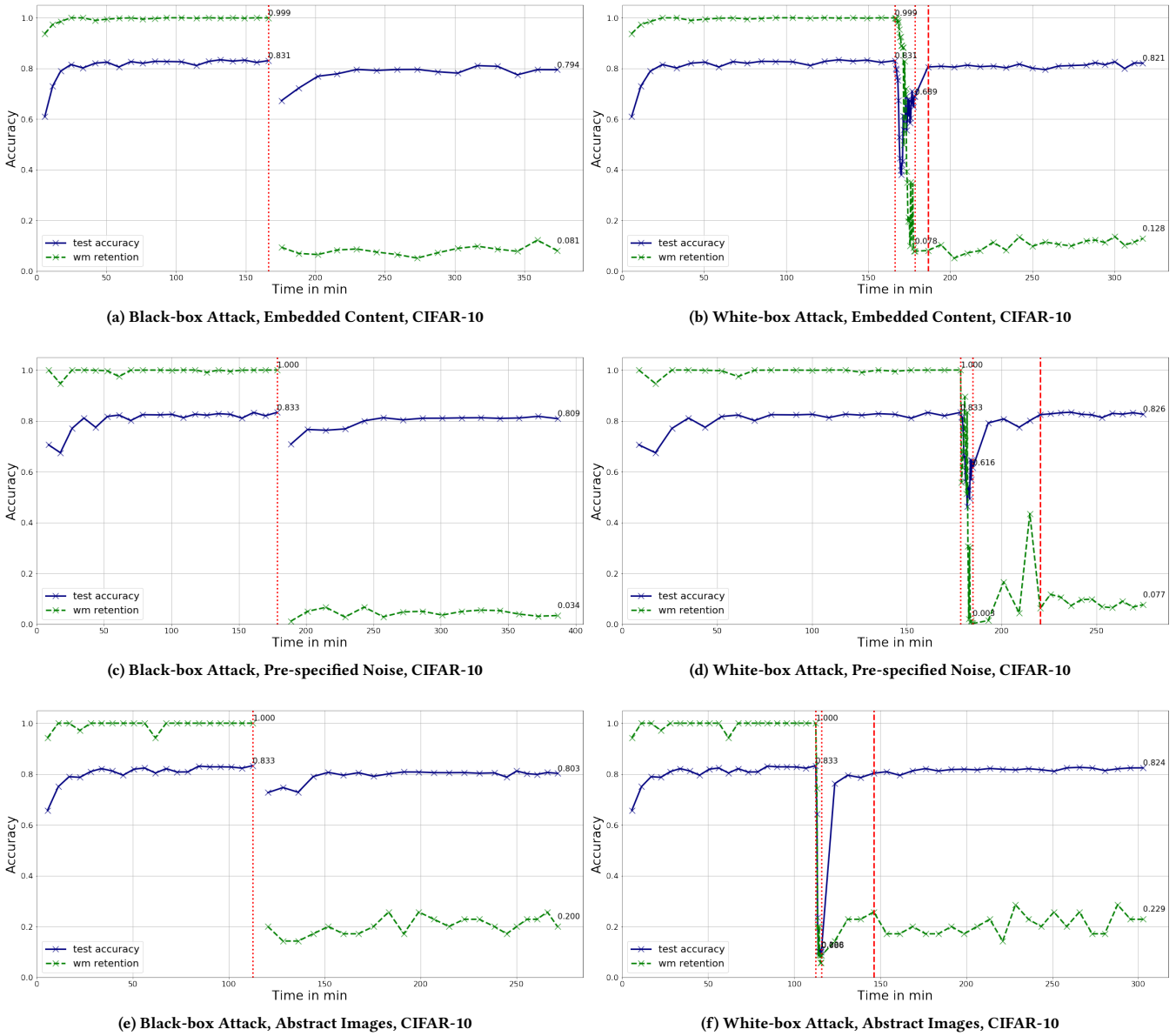
(e) Black-box Attack, Abstract Images, MNIST

(f) White-box Attack, Embedded Content, MNIST

Figure 8: Experiment results on MNIST for various backdoor-based watermarking schemes: (a,b) Embedded Content, (c,d) Pre-specified Noise, (e,f) Abstract Images, where (a), (c), (e) represent black-box attack results and (b), (d), (f) demonstrate the white-box attack results

than 10%(9.6%, 8.5%, 8.6%) while it causes a negligible drop in test accuracy(0.3%, 0.2%, 0.3%). The performance of the attack is comparable to the performance of the watermarked model(since we do not consider the extensive data preparation step for the watermarked model). The watermarked model takes 3.5, 2.6 and 2.6 minutes to train with the three mentioned watermarking schemes embedded, whilst our black-box attack takes 3.6, 4.1 and 5 minutes correspondingly to remove the watermark. The white-box attack graphs in Fig. 8(b), (d), (f) demonstrate significant speed-up compared to the black-box results. To remove the watermark while reaching the same test accuracy as our black-box attack, the white-box attack takes only 0.59, 0.77 and 2.1 minutes. However, we noticed that by continuing training the model in the white-box attack for a total time of 3.3, 3, 3.8 minutes for each watermarking scheme, we can even reach higher test accuracy values, i.e. 0.1% drop compared to the watermark model $\hat{M}$.

We evaluated our attacks on CIFAR-10 data as well. Fig. 9 (a), (c), and (e) show the results of our black-box attack on Embedded Content, Pre-specified Noise, and Abstract Images watermarking respectively, while subfigures (b), (d) and (f) indicate the corresponding



(a) Black-box Attack, Embedded Content, CIFAR-10

(b) White-box Attack, Embedded Content, CIFAR-10

(c) Black-box Attack, Pre-specified Noise, CIFAR-10

(d) White-box Attack, Pre-specified Noise, CIFAR-10

(e) Black-box Attack, Abstract Images, CIFAR-10

(f) White-box Attack, Abstract Images, CIFAR-10

Figure 9: Experiment results on CIFAR-10 for various backdoor-based watermarking schemes: (a,b) Embedded Content, (c,d) Pre-specified Noise, (e,f) Abstract Images, where (a), (c), (e) represent black-box attack results and (b), (d), (f) demonstrate the white-box attack results

results achieved by our white-box attack. The results show that our black-box attack reduces the watermark retention from nearly 100% in the watermarked model $\hat{M}$ to 8.1%, 3.4%, and 20% in $\tilde{M}$. The training time for $\hat{M}$ is 166, 178 and 112 minutes, whereas it takes the black-box attack 244, 208, and 158 minutes to train $\tilde{M}$. The required time could significantly be dropped to 20, 20, and 33 minutes by our white-box attack for the same accuracy. However, aiming for higher accuracy is also possible through our white-box attack since it results in 1%, 0.4%, and 0.9% test accuracy if it is permitted to train for 150, 123, and 190 minutes for each watermarking scheme.

Property Inference Attack- We investigated the effectiveness of our property inference attack to distinguish watermarked models from unmarked ones as shown in Fig. 10 (a) and (b). In (a) we trained a two-layer model M_{PI} by using 294 watermarked and unmarked models. The watermarked models were marked using Embedded Content method, and the tested M_{PI} on 33 models and the result is shown in Fig. 10. The network classifies watermarked and unmarked models perfectly after only 50 seconds of training. We repeated the experiment on models with Pre-specified Noise watermarking in (b). After training for 3.5 seconds with 113 watermarked

and unmarked models, our 2-layer model performs perfect classification. We leave further investigation of the probable reasons to the discussion.

5.3 Discussion

We provide further analysis of our attacks here and compare our work with the only other attack to backdoor-based watermarking schemes. We as well investigate deeper and present evidences that backdoor watermarking the model contradicts it achieving higher classification accuracy.

Full Removal of Watermark- As defined in Section 4, full watermark removal is achieved if the following two conditions are satisfied

(i) $Pr_{x\in D}[Classify(x, M) = f(x)] \approx Pr_{x\in D}[Classify(x, \tilde{M}) = f(x)]$

(ii) $\sum_{x\in T} \mathbb{I}[Classify(\tilde{M}, x) = T_L x] - \frac{1}{L}|T| \leq \epsilon|T|$

Our black-box attack removes the watermark with maximum $\epsilon = 0\%$ and $\epsilon = 20\%$ for MNIST and CIFAR-10 respectively, with corresponding test accuracy drop of $\tilde{M}$ compared to $\hat{M}$ is at maximum 0.3% and 3.7%. The attack takes up to almost twice as much time as the watermarked model training. We demonstrated the results of our black-box attack in combination with our proposed white-box attack. The white-box attack introduces two advantages over the black-box one. First, it removes the watermark while achieving the same accuracy as the black-box attack in considerably shorter time. It does so with maximum $\epsilon = 1.4\%$ for MNIST and $\epsilon = 13\%$ for CIFAR-10. Second, it can achieve higher accuracy(0.1% drop for MNIST and 1% drop for CIFAR) if trained for longer.

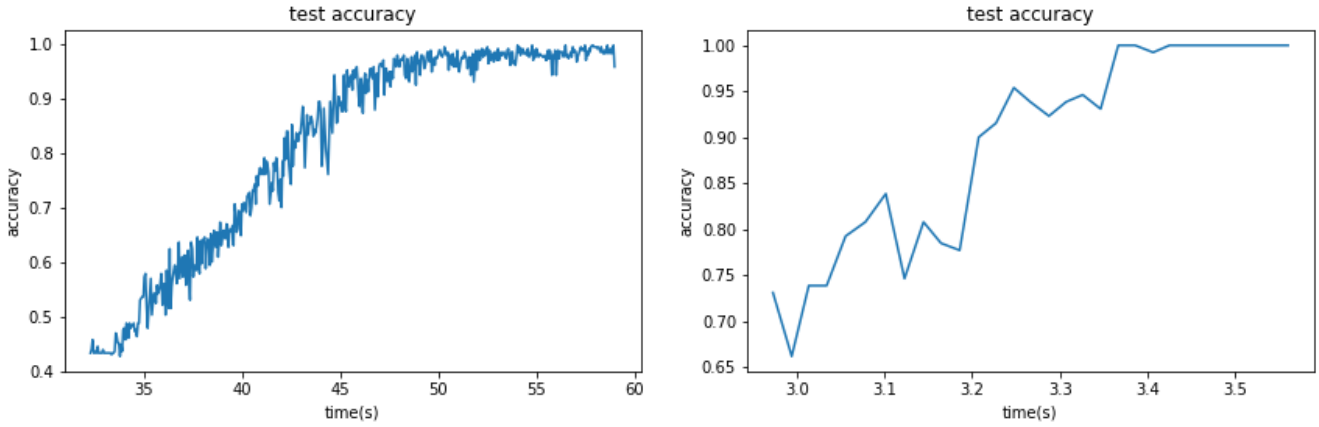
Comparison with Evasion attack- There has been just one other attack in DNN watermarking literature to remove backdoors [20], which steals n models and collects responses from all of them for each query. It then selects the answer that receives a higher vote among the responds from the stolen networks, and provides that as API prediction. The evasion attack has various disadvantages. First, it relies on accessing n models that perform the same task. It is also introduced as an online attack which requires availability of all n models for API prediction. Finally, it does not support any upgrade. In contrast, our attack does not require access to multiple networks to attack a watermarked network. We just query the target model for limited number of times, until we train our substitute model. Moreover, our substitute model can adapt to a more optimized version if it acquires more information, i.e. the watermarked model's parameters.

Watermark retention and test accuracy- In addition to the successful watermark removal by our attacks, we observed another important result in our experiments. As depicted in Fig. 11 for both MNIST and CIFAR-10 data sets in (a) and (b), our attacks on watermarked model $\hat{M}$ reach lower watermark retention if $\hat{M}$ achieves higher test accuracy. The reason is, the watermarked model $\hat{M}$ is the reference for $\tilde{M}$'s learning. If $\hat{M}$ does not achieve high accuracy, the inadequacy transfers to $\tilde{M}$ as well. Subsequently side factors would play a more important role in classifying the watermark test rather than the accurate model. We give an example of these "side factors" to clarify our argument. Considering the Content Embedded watermarking scheme, we know that the all elements in the watermark set, which has the size of 10% of the training data, are mapped to a pre-defined fixed class, e.g. class is 2. On the other hand, in classifying the rest of the training data we expect a balanced coverage of the labels, i.e. each label is mapped to approximately 10% of the training data. Now, the class 2 is mapped to 20% of the training data in total, resulting in an overall unbalanced class coverage. As a result, if a model $\hat{M}$ trained on this data is not accurate enough, it will carry the bias in classifications and transfers the bias to the subsequent network, $\tilde{M}$ as well. Therefore, $\tilde{M}$ is more resistant to watermark removal. Hence, what is implied in this observation is for a model to maintain higher watermark retention, it is encouraged to provide lower classification accuracy. This inherent contradiction impedes the backdoor-based watermarking schemes to fulfill their claim on the possibility of maintaining a high test accuracy and watermark retention at the same time.

Property Inference Attack- We demonstrated earlier that our property inference attack is capable of distinguishing the presence of a watermark in a given network. We provide further evidences in Fig. 12 to present the rational behind it. Fig. 12 indicates feature vectors for watermarked models(dashed line) and unmarked models(solid line). Fig. 12 (a) shows the weight average in the first layer of networks watermarked by Embedded Content approach and their difference with the weight averages of an unmarked model. As well, Fig. 12 (b) indicates the difference in the first layer biases of unmarked models and models watermarked with Pre-specified Noise. Adding the capability of detecting watermarks empowers our black-box and white-box attacks even more. First, for it enables them to target only the watermarked models. Second, it helps the attacks to perform more efficiently and accurately by providing them with a tool to distinguish the stopping point, instead of continuing training blindly for a fixed period of time.

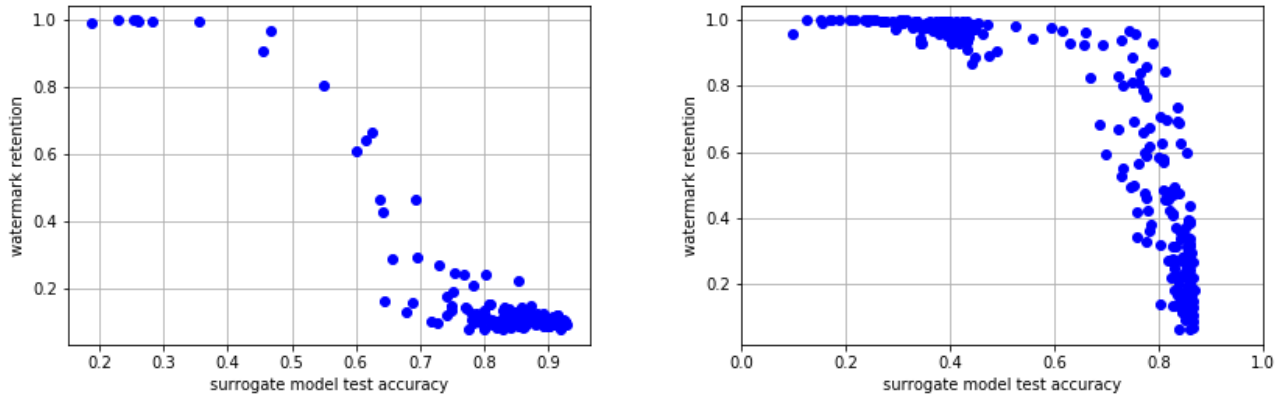
6 CONCLUSION

We presented three attacks on the recent backdoor-based watermarking schemes in deep neural networks; black-box attack, white-box attack, and property inference attack. The targeted schemes deploy the model's watermark in one the form of: logo devised by an embedded content, pre-specified noise pattern or trigger set consisting of abstract images. Our black-box and white-box attacks neither requires information about the type of embedded watermark, nor they need access to the ground truth function of the watermarked model, saving on enormous amount of time and resources required to prepare the training data. Instead, our attacks solely rely on the results of querying the watermarked model to label arbitrary inputs from the publicly known domain of the watermarked model. We should as well mention that since the total number of queries in either of our attacks in only $20,000 - 30,000$, it is difficult for the watermarked model to rely on approaches such as rate limiting to defend against them. We show that our attacks successfully remove the model watermark completely, with no sacrifice on classification accuracy of the model. Our black-box approach, is a surrogate model attack that accomplishes the full watermark removal task while knowing none of the watermarked model parameter and by solely exploiting its publicly available information. Although we show that granting more information, e.g. watermark model's parameters, facilitates devising a more



(a) Embedded Content **(b) Pre-specified Noise**

Figure 10: Property inference attack and watermark detection



(a) MNIST **(b) CIFAR**

Figure 11: Watermark retention versus test accuracy of the watermarked model $\hat{M}$ for (a) MNIST and (b) CIFAR

optimized attack. We introduced our white-box attack in this work which consists of regularization and fine-tuning processes and improves the black-box attack's performance noticeably. In addition to our two watermark removal attacks, we proposed a property inference attack that can distinguish a watermarked model from an unmarked one. This attack provides us with a powerful tool, to first recognize watermarked models and then apply our black-box or white-box attacks on. Second, it benefits our attacks by confirming when the watermark is fully removed and allows setting an accurate stop time to further improve our attacks' efficiency.

REFERENCES

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1615–1631. https://www.usenix.org/conference/usenixsecurity18/presentation/adi

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3 (March 2003), 1137–1155. http://dl.acm.org/citation.cfm?id=944919.944966

[3] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

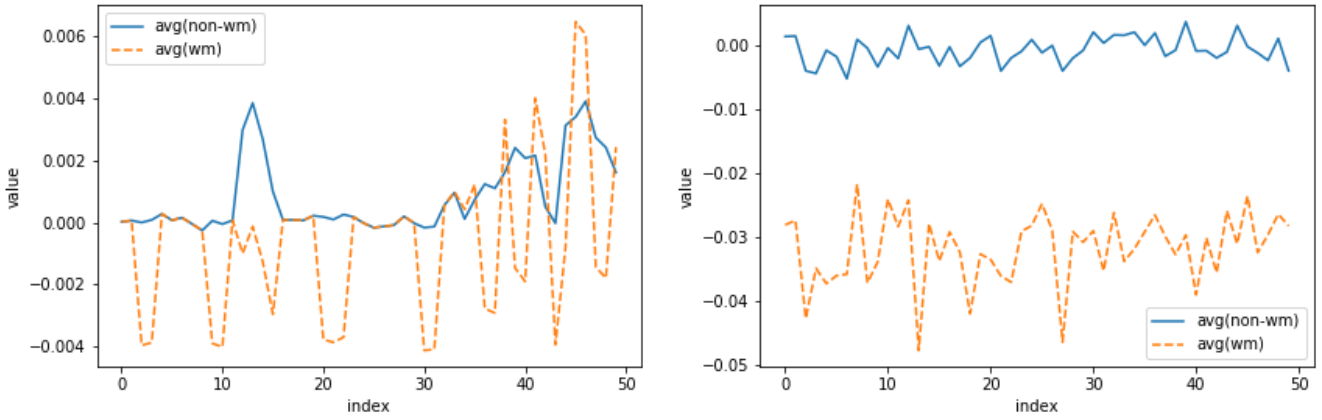
[4] Huili Chen, Bita Darvish Rohani, and Farinaz Koushanfar. 2018. DeepMarks: A Digital Fingerprinting Framework for Deep Neural Networks. *arXiv preprint arXiv:1804.03648* (2018).

[5] Huili Chen, Bita Darvish Rouhani, Xinwei Fan, Osman Cihan Kilinc, and Farinaz Koushanfar. 2018. Performance Comparison of Contemporary DNN Watermarking Techniques. *arXiv preprint arXiv:1811.03713* (2018).

[6] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. 2019. BlackMarks: Blackbox Multibit Watermarking for Deep Neural Networks. *arXiv preprint arXiv:1904.00344* (2019).

[7] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Xiaodong Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR* abs/1712.05526 (2017).

[8] Edward Chou, Florian Tramèr, Giancarlo Pellegrino, and Dan Boneh. 2018. SentiNet: Detecting Physical Attacks Against Deep Learning Systems. *arXiv preprint arXiv:1812.00292* (2018).



(a) First layer weight average in Embedded Content Watermarking

(b) First layer bias average in Pre-specified Noise Watermarking

Figure 12: Feature difference of watermarked and unmarked networks

[9] Crowd Flower. 2016. 2016 Data Science Report. (Mar 2016). Retrieved April 20, 2019 from https://visit.figure-eight.com/data-science-report.html

[10] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 619–633.

[11] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. STRIP: A Defence Against Trojan Attacks on Deep Neural Networks. *arXiv preprint arXiv:1902.06531* (2019).

[12] Yoav Goldberg. 2016. A Primer on Neural Network Models for Natural Language Processing. *J. Artif. Int. Res.* 57, 1 (Sept. 2016), 345–420. http://dl.acm.org/citation.cfm?id=3176748.3176757

[13] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. (02 2014).

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 38 (03 2013). https://doi.org/10.1109/ICASSP.2013.6638947

[15] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR* abs/1708.06733 (2017). arXiv:1708.06733 http://arxiv.org/abs/1708.06733

[16] Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[17] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep Speech: Scaling up end-to-end speech recognition. *CoRR* abs/1412.5567 (2014).

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.

[19] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *Signal Processing Magazine* (2012).

[20] Dorjan Hitaj and Luigi V Mancini. 2018. Have You Stolen My Model? Evasion Attacks Against Deep Neural Network Watermarking Techniques. *arXiv preprint arXiv:1809.00615* (2018).

[21] Mika Juuti, Sebastian Szyller, Alexey Dmitrenko, Samuel Marchal, and N. Asokan. 2018. PRADA: Protecting against DNN Model Stealing Attacks. *CoRR* abs/1805.02628 (2018). arXiv:1805.02628 http://arxiv.org/abs/1805.02628

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems* 25 (01 2012). https://doi.org/10.1145/3065386

[23] G.C. Langelaar, Iwan Setyawan, and R.L. Lagendijk. 2000. Watermarking digital image and video data. A state-of-the-art overview. *Signal Processing Magazine, IEEE* 17 (10 2000), 20 – 46. https://doi.org/10.1109/79.879337

[24] Zheng Li and Shanqing Guo. 2019. DeepStego: Protecting Intellectual Property of Deep Neural Networks by Steganography. *arXiv preprint arXiv:1903.01743* (2019).

[25] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. *CoRR* abs/1805.12185 (2018). arXiv:1805.12185 http://arxiv.org/abs/1805.12185

[26] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. https://doi.org/10.14722/ndss.2018.23300

[27] Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 45–48.

[28] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2017. Adversarial frontier stitching for remote neural network watermarking. *arXiv preprint arXiv:1711.01894* (2017).

[29] Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2018. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval* 7, 1 (01 Mar 2018), 3–16. https://doi.org/10.1007/s13735-018-0147-1

[30] Ryota Namba and Jun Sakuma. 2019. Robust Watermarking of Neural Network with Exponential Weighting. *arXiv preprint arXiv:1901.06151* (2019).

[31] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks Against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17)*. ACM, New York, NY, USA, 506–519. https://doi.org/10.1145/3052973.3053009

[32] Gil Press. 2016. Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. (Mar 2016). Retrieved April 20, 2019 from https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#77d71e046f63

[33] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2018. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750* (2018).

[34] Lalit Kumar Saini and Vishal Shrivastava. 2014. A Survey of Digital Watermarking Techniques and its Applications. *CoRR* abs/1407.4735 (2014).

[35] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556* (09 2014).

[36] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[37] Mitchell D. Swanson, Mei Kobayashi, and Ahmed Hossam Tewfik. 1998. Multimedia data-embedding and watermarking technologies. *Proc. IEEE* 86, 6 (1 12 1998), 1064–1087. https://doi.org/10.1109/5.687830

[38] Florian Kerschbaum Tianhao Wang. 2019. Attacks on Digital Watermarks for Deep Neural Networks. In *44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

[39] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs. In *Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16)*. USENIX Association,

Berkeley, CA, USA, 601–618. http://dl.acm.org/citation.cfm?id=3241094.3241142

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

[41] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks*. IEEE, 0.

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3320–3328. http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

[43] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, Edwin R. Hancock Richard C. Wilson and William A. P. Smith (Eds.). BMVA Press, Article 87, 12 pages. https://doi.org/10.5244/C.30.87

[44] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 159–172.