SoK: How Robust is Image Classification Deep Neural Network Watermarking?

Nils Lukas, Edward Jiang, Xinda Li, Florian Kerschbaum
University of Waterloo
Waterloo, Canada
{nlukas, eydjiang, xinda.li, florian.kerschbaum}@uwaterloo.ca

2022 IEEE Symposium on Security and Privacy (SP) | 978-1-6654-1316-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SP46214.2022.9833693

***Abstract*—Deep Neural Network (DNN) watermarking is a method for provenance verification of DNN models. Watermarking should be robust against watermark removal attacks that derive a *surrogate* model that evades provenance verification. Many watermarking schemes that claim robustness have been proposed, but their robustness is only validated in isolation against a relatively small set of attacks. There is no systematic, empirical evaluation of these claims against a common, comprehensive set of removal attacks. This uncertainty about a watermarking scheme's robustness causes difficulty to trust their deployment in practice. In this paper, we evaluate whether recently proposed watermarking schemes that claim robustness are robust against a large set of removal attacks. We survey methods from the literature that (i) are known removal attacks, (ii) derive surrogate models but have not been evaluated as removal attacks, and (iii) novel removal attacks. *Weight shifting* and *smooth retraining* are novel removal attacks adapted to the DNN watermarking schemes surveyed in this paper. We propose taxonomies for watermarking schemes and removal attacks. Our empirical evaluation includes an ablation study over sets of parameters for each attack and watermarking scheme on the image classification datasets CIFAR-10 and ImageNet. Surprisingly, our study shows that none of the surveyed watermarking schemes is robust in practice. We find that schemes fail to withstand adaptive attacks and known methods for deriving surrogate models that have not been evaluated as removal attacks. This points to intrinsic flaws in how robustness is currently evaluated. Our evaluation includes a discussion of the runtime of each attack to underpin their practical relevance. While none of the schemes is robust against all attacks, none of the attacks removes all watermarks. We show that attacks can be combined and find combined attacks that remove all watermarks. We show that watermarking schemes need to be evaluated against a more extensive set of removal attacks with a more realistic adversary model. Our source code and a complete dataset of evaluation results are publicly available, which allows to independently verify our conclusions.**

***Index Terms*—Deep Neural Network, Watermarking, Robustness, Removal Attacks, Image Classification**

I. Introduction

Deep Neural Networks (DNN) have become state-of-the-art algorithms for applications such as facial recognition [1]–[3], medical image classification [4] and autonomous driving [5]. Training a DNN model can be expensive due to data preparation (collection, organizing, and cleaning) and computational resources required for validating a model [6]. For this reason, DNNs are often provided by a single entity and consumed by many, such as in Machine Learning-as-a-Service (MLaaS). A model provider may want to restrict unauthorized redistribution of their *source* model. The threat to the model provider is a user who derives a (stolen) *surrogate* model from access to the source model and publicly deploys their surrogate model. Krishna et al. [7] have shown that such *model stealing* attacks can be (i) effective because even high-fidelity surrogates of large models like BERT [8] can be derived with limited access to domain data and (ii) practical because surrogate models can be derived for a fraction of the costs compared to retraining a model.

Papernot et al. [9] describe the *confidentiality* requirement as one of the core principles for security and privacy in machine learning. Preserving a model's confidentiality refers to protecting its parameters against model stealing attacks. Confidentiality is important because the source model constitutes intellectual property and may leak information about its training dataset. Preventing model stealing is difficult [7], [10]–[12], but detecting whether the confidentiality of a source model has been broken serves as a powerful deterrent and can be achieved through DNN *watermarking*.

DNN watermarking [13] is a method designed to detect surrogate models. Watermarking embeds a message into a model that is later extractable using a secret key. Developing DNN watermarking schemes is an active area of research studied by large corporations such as Microsoft [14], Google [15] and IBM [16]. Robustness is a core security property of watermarking, which states that an attacker cannot derive surrogate models from access to the source model that do not retain the watermark. Watermarking schemes that are robust against such *watermark removal* attacks are needed to deter redistribution by adversaries. Claimed security properties of some existing watermarking schemes [15], [16] had been broken by novel attacks [17]–[19], but it is unclear how these attacks generalize to other watermarks.

We perform a systematic evaluation and propose taxonomies for watermarking schemes and attacks. We survey 29 methods from the literature that (i) are known removal attacks, such as weight pruning [20] or knowledge distillation [21], (ii) derive surrogate models but have not been evaluated as removal attacks, and (iii) novel removal attacks. A removal attack is *effective* if the surrogate model has a high test accuracy and does not retain the watermark. It is *efficient* if resources required to run the attack, such as its runtime, are small compared to retraining a model from scratch. We measure both effectiveness and efficiency. In our taxonomy, we categorize attacks into (i) model modification, (ii) input preprocessing,

DOI 10.1109/SP46214.2022.00004

and (iii) model extraction. Model modification and input preprocessing modify the source model or its input, whereas model extraction trains a different surrogate model by distilling knowledge from the source model.

We survey eleven[1] recently proposed watermarking schemes [13]–[16], [22]–[26] from the literature that claim robustness. Most of these schemes do not specify whether their definition of robustness includes model extraction [13], [14], [16], [22], [23], one scheme restricts the runtime of the attacker [15] and the remaining schemes claim robustness against any removal attack [24]–[26]. In this paper, we evaluate robustness against any removal attack and demonstrate whether an attack is efficient by showing its runtime. Our taxonomy categorizes these watermarking schemes into (i) model independent, (ii) model dependent, (iii) parameter encoding, and (iv) active watermarking schemes.

Our new Watermark-Robustness-Toolbox (WRT) implements all watermarking schemes and removal attacks evaluated in this paper. We validate the robustness of each scheme against each removal attack. Our evaluation includes an ablation study over multiple sets of parameters for each watermarking scheme and removal attack. The defender and attacker engage in a zero-sum game to choose the best parameter set for their method, which constitutes the Nash equilibrium. We say a scheme is robust if the defender can choose a set of parameters so that no removal attack is effective. Our study analyzes the robustness of watermarking schemes and the effectiveness and efficiency of removal attacks. We also study the robustness of watermarking scheme categories against categories of removal attacks to identify the category of most effective attacks that should be used to evaluate the robustness of a watermarking scheme in a specific category.

Our empirical evaluations are performed on large datasets to emphasize the practical relevance of our work. The experiments span CIFAR-10 [27] and ImageNet [28], which are image classification datasets. The ImageNet dataset contains over 1.2 million training images from 1k categories and is a broadly accepted benchmark to measure the performance of state-of-the-art machine learning models [29].

Our study shows that none of the investigated watermarking schemes is robust against all removal attacks. However, we also find that none of the attacks from the literature removes all watermarks. We propose new *combined* attacks that remove all investigated watermarks while maintaining a high test accuracy in the surrogate models. Our study also shows that robustness should be verified against a more extensive set of attacks and on a larger number of datasets. We believe that an open-source implementation of watermarking schemes and removal attacks enhances the scientific study of a scheme's robustness. Towards this goal, we make our new Watermark-Robustness-Toolbox (WRT) and a complete dataset of evaluation results publicly available with documentation, which allows independently verifying our conclusions.

[1]Zhang et al. [16] propose three different schemes.

Requirements	Description
Fidelity	The impact on the model's task accuracy is small.
Robustness	Surrogate models retain the watermark.
Integrity	Models trained without access to the source model do not retain the watermark.
Capacity	The watermark allows encoding large messages sizes.
Efficiency	Embedding and extracting the watermark is efficient.
Undetectability	The watermark cannot be detected efficiently without knowledge of the secret watermarking key.

TABLE I: Requirements for ideal DNN watermarking.

A. Contributions

This work contributes:

- Taxonomies of DNN watermarking schemes and removal attacks.
- An empirical evaluation of the robustness of DNN watermarking schemes [13]–[16], [22]–[26] against removal attacks from related work.
- A unified adversary model for the attacker and defender in any of the evaluated watermarking schemes.
- Proposal of the novel removal attacks *weight shifting* and *smooth retraining*.
- Combined attacks that remove all surveyed watermarks.
- Guidelines to evaluate the robustness of watermarking.
- An open-source implementation of all watermarking schemes and removal attacks evaluated in this paper.

B. Organization

The rest of the paper is organized as follows. Section II describes background information on deep neural networks. Section III presents our taxonomy on watermarking schemes and removal attacks and Section IV describes a unified adversary model for the attacker and defender. Section V presents the methodology for our experiments and defines all measured quantities for our experiments. Empirical results are presented in Section VI. Section VII-A presents guidelines for evaluating robustness and Section VIII concludes the paper. Descriptions of the watermarking schemes and attacks and parameters for our ablation study can be found in Appendix X and XI.

An extended version of this paper is available as a technical report [30]. This report additionally contains survey-style descriptions of the investigated watermarking schemes and removal attacks and an extended discussion on their weaknesses and strengths.

II. Background

A. Deep Neural Networks (DNNs)

A deep neural network (DNN) classifier is a function $M : \mathcal{X} \to \mathcal{Y}$ that assigns a likelihood to inputs $\mathcal{X} \subseteq \mathbb{R}^d$ for each of $K \in \mathbb{N}$ classes $\mathcal{Y} \subseteq \mathbb{R}^K$. It is a sequence of layers $f_i, (i \in \{1, .., L\})$ in which each layer implements a linear function followed by a non-linear function called the activation function. A neural network is called deep if it has more than one layer between the input and output layer, called hidden layers. Hidden layers have weight and bias parameters used to compute that layer's activations. A softmax activation

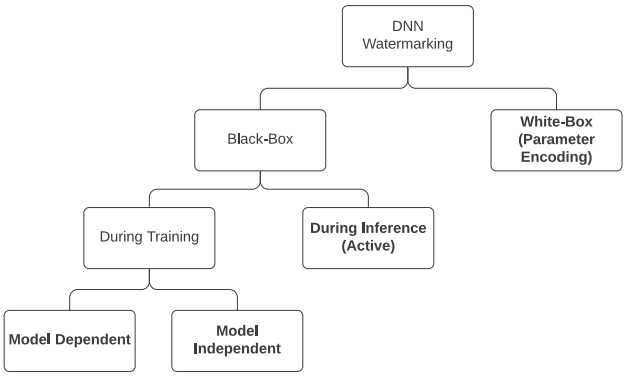

Fig. 1: A categorization of watermarking schemes. The distinction between 'white-box' and 'black-box' refers to the verification requirement, whereas 'during training' and 'during inference' refer to the embedding of the watermark.

function $\sigma(\cdot)$ is applied to the output layer $f_L(\cdot)$ to convert likelihoods into probabilities for each predicted class.

$$\sigma(f_L(x))_i = \frac{\exp(f_L(x)_i)}{\sum_j \exp(f_L(x)_j)} \tag{1}$$

Training a neural network model requires the specification of a differentiable loss function that is optimized by gradient descent on all trainable weights and biases. One such loss function is the cross-entropy loss H for some ground truth $y \in \mathcal{Y}$ with respect to the model's prediction.

$$H(y, f_L(x)) = -\sum_{0 \leq k < K} (y_k \cdot \log(\sigma(f_L(x))_k)) \tag{2}$$

A black-box deployment of a DNN exposes only the API of the model. On input of an element $x \in \mathcal{X}$, the server responds with the full confidence vector $\sigma(f_L(x)) \in \mathcal{Y}$.

III. Taxonomy of Watermarking

In this section, we define DNN watermarking and describe our proposed taxonomy. We introduce watermarking as a method for DNN provenance verification and propose categorizations of watermarking schemes and removal attacks.

A. Defining Watermarking.

Watermarking embeds a message into a *source model* that is later extractable using a secret watermarking key. The *success rate* between two messages can be computed as the number of matching bits normalized by the message length. It is defined as follows for messages $a, b \in \{0,1\}^N$ of size N, where δ denotes the Kronecker delta.

$$\Delta(a, b) = \frac{1}{N} \sum_{i=1..|N|} \delta(a_i, b_i)$$

A watermark is *retained* in a model if the same message can be extracted with a success rate that is higher than a *decision threshold*, defined by the watermarking scheme. Otherwise, we say that a watermark is removed. A watermark should be retained in *surrogate models* that are derived from the source model. Methods of derivation include modifying the source model, e.g., through fine-tuning [13] or weight pruning [20], and extraction of the source model, which uses a process related to knowledge distillation [21] to train a different model.

We differentiate between zero-bit and multi-bit watermarking. Zero-bit watermarking encodes only the presence of a watermark, whereas multi-bit watermarking allows encoding a message containing several bits of information. For zero-bit watermarking schemes, we represent the message that can be extracted as a bit-string $m \subset \{0,1\}$, where $m_i = 1$ means that the presence of the i-th watermark has been detected and $m_i = 0$ otherwise. Note that the message embedded into the source model has zero bits because extracting from a source model after embedding should always return the message of all ones. For multi-bit watermarking, the message $m \subset \{0,1\}$ can be chosen by the user and thus contains multiple bits of information. A watermarking scheme can be formalized by an embedding and extraction procedure.

- **Embed**(T, m, M): Takes a watermarking key T, a message $m \subset \{0,1\}$ and a model M and outputs a marked model $\hat{M}$ embedded with a message m.
- **Extract**(T, M): Takes a watermarking key T, a model M and outputs the message $m \subset \{0,1\}$ extracted from model M using key T.

The watermarking key T contains the secret information required to extract a watermark. For example, the watermarking key can consist of images [15], a bit-vector [13] or a combination of both [14]. There exists a trivial procedure to verify whether a model $\hat{M}$ retains a watermark. This verification procedure takes as parameters a watermarking key T and message m, a model $\hat{M}$ and a decision threshold $\theta \in [0,1]$. The decision threshold specifies the lowest tolerable success rate between message m and the message extracted from $\hat{M}$ to verify whether the watermark is retained. The verification extracts a message $\hat{m}$ from model $\hat{M}$ using T and computes the success rate $d = \Delta(\hat{m}, m)$. If the watermark is retained ($d \geq \theta$) the verification outputs $b = 1$ and $b = 0$ otherwise.

B. Watermarking Categories

We systematize DNN watermarking schemes as a tree diagram in Figure 1. These schemes can be differentiated by (i) the watermark carrier, (ii) the stage at which the watermark is embedded, and (iii) whether the embedding requires access to a pre-trained source model for the generation of the watermarking key. The watermark carrier can be the model's parameters or its functionality. In the latter case, modification of the functionality can either occur during inference while the model is deployed or during training. If the embedding occurs during training, a watermarking scheme may require that the model is pre-trained. In this case, the secret key's generation depends on the trained model, e.g., when the secret key contains adversarial examples [31]. Otherwise, the watermarking key can be generated independently of the model and only depends on the dataset. In summary, for a systematic analysis

of the robustness of watermarking, we propose the following four categories of watermarking schemes.

1) **Model Dependent** [23]–[25]: A model dependent scheme embeds the message into the model's functionality during training, where the watermark key samples depend on the model. Watermarking schemes in this category either rely on adversarial examples [32] generated for the source model [23], [25] or use the source model to modify the watermarking key [24].
2) **Model Independent** [15], [16], [33]: A model independent scheme embeds the message into the functionality during training, where the watermarking key samples do not depend on the model. The watermark is a *backdoor* [34], i.e., secret functionality learned by the source model from the training set. A backdoor is embedded by injecting additional samples into the training set, and hence, the generation of the watermarking key does not depend on the source model.
3) **Active** [26]: An active scheme embeds the message into the model's functionality during inference. It requires the defender to control the source model's deployment. Active schemes only defend against attackers with black-box access to the source model by *postprocessing* predictions returned by the source model on input queries.
4) **White-box** (Parameter Encoding) [13], [14], [22]: A *white-box* scheme embeds the message into the model's parameters [13], [22] or into the activations of its hidden layers [14]. Verification requires white-box access to the source model, i.e., access to the model's parameters.

C. Watermark Removal Attack Categories

A watermark removal attack takes as input the source model and outputs a surrogate model. It is successful if the surrogate model does not retain the watermark, and it has a similar utility (measured in test accuracy) as the source model. We survey (i) known removal attacks [13], [17], [20], [35]–[38], (ii) methods that derive a surrogate model but have not been evaluated as removal attacks against DNN watermarking [21], [39]–[43], [43]–[47] and (iii) novel, adaptive attacks proposed in this paper. We investigate which of these methods successfully remove watermarks. From all surveyed removal attacks, we derive the following three attack categories.

- **Input Preprocessing**: Input preprocessing attacks modify the data samples for classification before passing them through the surrogate model. The attacker must have white-box access to the source model.
- **Model Modification**: Model modification attacks transform the source model's parameters, e.g., by fine-tuning [13] or pruning [20]. The attacker must have white-box access to the source model.
- **Model Extraction**: Model extraction attacks train a different surrogate model by transferring knowledge from the source model into the surrogate model. The surveyed model extraction attacks need only black-box access to the source model, with the exception of knowledge distillation [21] which requires white-box access.

D. Formalizing Watermarking Requirements

Ideal watermarking should satisfy the requirements listed in Table I. We now formalize the two properties investigated in this paper: robustness and integrity. We refer to the watermark extraction procedure by $E(T, \hat{M})$ for ease of notation.

Robustness: Robustness requires that a message extracted from a surrogate model is approximately the same as the message extracted from the source model. The following condition should hold for $\varepsilon \geq 0$, a model M, a watermarking key T, a message m and any watermark removal attack $\mathcal{A}$.

$$\hat{M} \leftarrow \text{Embed}(T, m, M)$$
$$\Delta\big(E(T, \hat{M}), E(T, \mathcal{A}(\hat{M}))\big) \geq 1 - \varepsilon$$

Note that robustness as defined is trivial by itself for zero-bit watermarking since the extraction algorithm could always return an all-ones message.

Integrity: Integrity requires a low success rate between messages extracted from a marked model $\hat{M}$ and an unmarked model M_0. Given the watermarking key T, a message m, the marked model $\hat{M}$ as defined above and an unmarked model M_0 the following condition should hold for $\varepsilon \geq 0$.

$$\Delta(E(T, \hat{M}), E(T, M_0)) \leq \varepsilon$$

We evaluate whether DNN watermarking can satisfy robustness and integrity. In the next section, we define a generic adversary model and present all watermarking schemes and removal attacks evaluated in this paper.

IV. Adversary Model

In this section, we describe the attacker's goals and capabilities. Our study covers many different watermarking schemes and removal attacks that assume different adversary models. For example, model modification attacks require white-box access to the source model, whereas many model extraction attacks only require black-box access. We present a generic adversary model for any watermarking scheme and watermark removal attack. Tables II and III summarize the defender's and attacker's capabilities for all methods surveyed in this paper.

A. Attacker's Goals

The attacker's primary goal is to derive a surrogate model from access to the source model (i) without the retained watermark that is (ii) *well-trained*, i.e., it has a similar test accuracy as the source model. A secondary goal is to reduce resources needed for the removal attack, such as the attack's computation time. We formalize a security game between the attacker and the defender. Given a secret watermarking key T and message m, only known to the defender, two well-trained, unmarked models M, M_0 and a watermark removal attack $\mathcal{A}$, the security game can be formalized as follows for $\varepsilon \geq 0$.

1) Train M and M_0 and send M to the defender.
2) Defender embeds the watermark $\hat{M} \leftarrow \text{Embed}(T, m, M)$
3) Attacker derives the surrogate model $M_1 \leftarrow \mathcal{A}(\hat{M})$
4) Sample $M_b \xleftarrow{\$} \{M_0, M_1\}$ and send M_b to the defender

Defense	Category	Verification	Capacity
Adi [15]	Model Independent	Black-Box	Multi-bit
Content [16], Noise [16], Unrelated [16]	Model Independent	Black-Box	Zero-bit
Jia [24], Frontier Stitching [25]	Model Dependent	Black-box	Zero-bit
Blackmarks [23]	Model Dependent	Black-box	Multi-bit
Uchida [13], Deepsigns [14], DeepMarks [22]	Parameter Encoding	White-box	Multi-bit
DAWN [26]	Active	Black-box	Multi-bit

TABLE II: All watermarking schemes evaluated in this paper. See Appendix X for a description of each method.

Attack	Category	Deployment	Data
Input Reconstruction [46], JPEG Compression [44], Input Quantization [42], Input Smoothing [43], Input Noising [45], Input Flipping, Feature Squeezing [43]	Input Preprocessing	White-box	None
Adversarial Training [41], Fine-Tuning (RTLL, RTAL) [13], Weight Quantization [47], Label Smoothing [48], Fine Pruning [38], Feature Permutation (Ours), Weight Pruning [20], Weight Shifting (Ours), Neural Cleanse [37], Regularization [17], Neural Laundering [35]	Model Modification	White-box	Domain
Overwriting [13], Fine-Tuning (FTLL, FTAL) [13]	Model Modification	White-box	Labeled
Knockoff Nets [40]	Model Extraction	Black-box	Transfer
Distillation [21]	Model Extraction	White-box	Domain
Transfer Learning, Retraining [36], Smooth Retraining (Ours) Cross-Architecture Retraining (Ours), Adversarial Training (From Scratch) [41]	Model Extraction	Black-box	Domain

TABLE III: A list of all watermark removal attacks evaluated in this paper and the attacker's capabilities (see Section IV). We refer to Appendix XI for a more detailed description of the attacks and their parameters used for our ablation study. RTAL and RTLL use predicted labels, whereas FTAL and FTLL use ground-truth labels (otherwise, gradients are zero).

5) Attacker wins if:

$$\Pr[\text{Verify}^2(T, M_b) = b] \leq 0.5 + \varepsilon$$

The robustness and integrity of a watermarking scheme are violated if an attacker can win this security game.

B. Attacker's Capabilities.

We now present the capabilities of an attacker in the form of a unified adversary model. Tables II and III summarize the adversary model for each watermarking scheme and removal attack surveyed in this paper.

Deployment. The deployment property summarizes the access of the attacker to the source model's parameters. It is white-box if all of the source model's parameters are accessible to the attacker and black-box if only the source model's API is accessible. Note that an attacker with white-box access is more informed and can also invoke attacks of an attacker who only has black-box access.

Dataset. The dataset property summarizes the availability of an auxiliary dataset to the attacker. Many attacks from related work require at least the availability of unlabeled domain data, and some even need access to data where a subset is labeled with ground-truth labels. We assume the attacker is limited in the amount of labeled data; otherwise, they could train their own model and would not need to steal the defender's source model. From all attacks, we identify the availability of the following three datasets to the attacker.

1) **Labeled**: Data from the same distribution where a subset of at most a third of the data is labeled.
2) **Domain**: Unlabeled data from the same distribution.
3) **Transfer**: Labeled data from a different distribution.

An attacker with access to a subset of labeled data is more informed than an attacker with access to only domain data. We consider collecting labeled data from a different distribution, and in all of our experiments, we use the Open Images [49] dataset as our transfer set.

Speed. Throughout the paper, we assume unbounded computational resources for the attacker. We only measure the runtime of attacks for a discussion of the practicality of the attack. Attacks are categorized concerning the total training time of an unmarked model from scratch. We consider an attack to be *fast* if it requires less than 25% of the training time, *medium* for times between 25% and 75% and *slow* for longer runtimes. We categorize speed according to the attack's runtime on the highest resolution dataset investigated in this paper (i.e., ImageNet [28]).

V. Measured Quantities

In this section, we present the measured quantities for conducting our experiments and describe the criteria for a watermark to be considered robust. Quantities, such as the test accuracy or an attack's runtime, are measured for the outcome of each removal attack against every watermarking scheme. We describe a method to empirically determine a decision threshold (see Section II) for each watermarking scheme and dataset. We introduce the *Nash equilibrium* as a method to determine the best choice of parameters in an adversarial setting. The Nash equilibrium is computed over multiple parameter configurations for each scheme and removal attack. Our goal is to empirically determine whether watermarking schemes are robust to removal attacks.

[2]The process 'Verify' checks if the success rate of the embedded and extracted message is higher than the decision threshold (see Section III-A)

A. Measurements

First, we describe the quantities measured for each experiment and our processing of these measurements to ensure comparability between watermarking schemes.

Embedding and Stealing Losses. We measure the *embedding* and *stealing losses* as differences in test accuracy between an unmarked and a marked model and between a marked and a stolen surrogate model. The test accuracy is the accuracy of a model's predictions on an unseen, labeled dataset from the same distribution. First, we define an auxiliary function that computes the accuracy of a model M on a dataset $D \subseteq \mathcal{X} \times \mathcal{Y}$.

$$\text{acc}(M, D) = \Pr_{(x,y)\in D}[\arg\max_i(M(x)) = \arg\max_j(y)]$$

The embedding loss is the difference in test accuracy between an unmarked model M_0 and a marked source model $\hat{M}$ on a labeled test dataset $D_{val} \subseteq \mathcal{X} \times \mathcal{Y}$.

$$L_{embed}(M_0, \hat{M}, D_{val}) = \text{acc}(M_0, D_{val}) - \text{acc}(\hat{M}, D_{val})$$

The stealing loss is the difference in test accuracy between a marked source model $\hat{M}$ and a stolen surrogate model M_S.

$$L_{steal}(\hat{M}, M_S, D_{val}) = \text{acc}(\hat{M}, D_{val}) - \text{acc}(M_S, D_{val})$$

The defender wants to minimize the embedding loss and the attacker wants to minimize the stealing loss.

Watermark Accuracy. The watermark accuracy is equal to the success rate defined in Section III-A. We define the watermark accuracy for a surrogate model $\hat{M}$ and the message m embedded into the source model using the secret watermarking key T. Let E be the message extraction function described in Section III-D.

$$\text{wmacc}(\hat{M}, m) = \Delta(E(T, \hat{M}), m)$$

Decision Threshold. The decision threshold $\theta \in [0, 1]$ determines the lowest tolerated watermark accuracy to verify that a watermark is retained in a model. Ideally, a scheme defines a decision threshold as part of their adversary model that we could use to assess its robustness. Unfortunately, such methods are missing from the surveyed papers, meaning that we have to find a methodology to empirically derive decision thresholds for each watermarking scheme.

Determining the decision threshold for a watermarking scheme is difficult. The decision threshold depends on the watermark accuracy of an unmarked model, which can be influenced by factors such as the model's architecture or the randomness during training. For example, consider the case of the zero-bit, model dependent watermarking scheme Frontier Stitching [25]. The presence of a watermark is detected if a surrogate model predicts the ground-truth labels for images that are part of the watermarking key. The watermarking key is composed of adversarial examples [32] generated for the source model. During the embedding, the source model is adversarially trained [41] to predict ground-truth labels for the watermarking key, whereas unmarked models still likely predict incorrect labels if the adversarial examples are transferable [50]. The problem is that the watermark accuracy of an unmarked model can increase without access to the source model by using adversarial training. This affects this watermarking scheme's decision threshold, which should be chosen large enough so that unmarked models are not incorrectly verified. The challenge lies in estimating the cumulative probability distribution that an unmarked model has a watermarking accuracy larger than some decision threshold. Such an estimation enables determining a decision threshold so that an incorrect verification (i.e., falsely claiming that a watermark is retained in a model) has a given probability.

Modeling the Decision Threshold. We empirically estimate an unmarked model's watermark accuracy given two random variables: the unmarked model and the watermarking key. Our goal is to estimate the cumulative probability that the watermark accuracy of a randomly generated watermarking key and a randomly sampled unmarked model is higher than some threshold. We make an i.i.d. assumption for our random variables and randomly generate 100 watermarking keys, each with a bit-length of $N = 100$. Then, we compute the watermark accuracy on a set of 30 unmarked models for CIFAR-10 and 20 unmarked models for ImageNet for every key and model pair. We model the cumulative normal probability distribution for the expected number of matched bits and choose a decision threshold. For our experiments, we choose a p-value of 0.05. Table IV shows a summary of the resulting decision thresholds for CIFAR-10 and ImageNet. We observe that some decision thresholds are different between CIFAR-10 and ImageNet, which requires the defender to derive a threshold specific to the model and dataset they want to protect. For the watermarking schemes Content, Noise, Frontier Stitching and Blackmarks, we observed that the choice of parameters affects their decision thresholds. In these cases, Table IV shows the largest computed decision threshold, and we refer to Appendix XIII for more information.

Rescaling Watermark Accuracies. Our goal is to compare the robustness of different watermarking schemes. Relating watermark accuracies from different schemes with each other is difficult because their decision threshold may differ. In such cases, the watermark accuracy alone does not indicate whether a scheme is robust without knowledge of the scheme's decision threshold. We avoid this issue by linearly rescaling the watermark accuracy by the scheme's decision threshold θ so that a watermark is retained if the *rescaled* watermark accuracy is at least equal to some fixed value $\theta' = 0.5$ and removed otherwise. This allows us to plot the watermark accuracies for different schemes into the same graph. We define a linear scaling function $S(x; \theta)$ that rescales the watermark accuracy so that (i) $S(\theta; \theta) = \theta'$ and (ii) $S(1; \theta) = 1$. The rescaling function uses the scheme's (unscaled) decision threshold θ as a parameter and returns the scaled watermark accuracy.

$$S(x; \theta) = \max(0, \frac{1 - \theta'}{1 - \theta}x + \frac{\theta' - \theta}{1 - \theta}) \tag{3}$$

We clip the output to avoid negative watermark accuracies. From this point forward, unless stated otherwise, we only refer to the rescaled watermark accuracy and decision threshold.

	Content	Noise	Unrelated	Adi	Jia	FS	Blackmarks	Deepmarks	Deesigns	Uchida	Dawn
CIFAR-10	0.0717	0.4867	0.1485	0.1504	0.0518	0.5330	0.6225	0.3964	0.5254	0.5798	0.1641
ImageNet	0.0018	0.0229	0.0074	0.0066	0.1638	0.7164	0.8073	0.3183	0.5848	0.5817	0.0061

TABLE IV: This table shows the empirically determined, unscaled decision thresholds for each watermarking scheme on two datasets with a p-value of 0.05. We obtain these decision thresholds by generating 100 watermarking keys with a key length of $N = 100$ each and compute the mean watermark accuracy on a set of unmarked models. We use 30 unmarked models for CIFAR-10 and 20 models for ImageNet. We refer to Appendix XIII for details on the computation of the decision thresholds.

Runtime. The runtime helps assess the practicality of a watermarking scheme or removal attack. We measure the runtime to (i) embed the watermark and (ii) run a removal attack. Since runtimes depend on the hardware, we report all runtimes measured on (single) Tesla P100 GPUs.

Attack Success Criterion. A success criterion determines whether a removal attack was successful in removing a watermark. We consider the watermark accuracy and the stealing loss of the surrogate model. We say a removal attack was successful when the surrogate model's watermark accuracy is lower than the scheme's decision threshold and the surrogate model is well-trained. In our paper, we consider a maximum stealing loss of *five* percentage points for a surrogate model to be considered well-trained. We refer to Section IV-A for a security game that formalizes our success criterion.

B. Nash Equilibrium

Our empirical analysis performs an ablation study over multiple sets of parameters for each watermarking scheme and removal attack. We now describe a method to measure the robustness of a watermarking scheme against one or more removal attacks under the consideration that the defender and attacker can choose from a set of parameters. For every watermarking scheme and removal attack, we ablate over multiple parameters (see Appendix X and XI) from which the defender and attacker can choose. We define a zero-sum game between the defender and attacker, where both players want to choose optimal parameters to maximize their gains.

We construct a *payoff* matrix $V \in \mathbb{R}^{m \times n}$ for n watermarking scheme parameters $\{d_0, .., d_n\}$ and m removal attack parameters $\{a_0, .., a_m\}$. The defender and attacker have full knowledge of this payoff matrix. An entry in this matrix is computed by applying a payoff function on the outcome of running an attack with the row's parameters against a watermarking scheme with the column's parameters. We define the following payoff function. The payoff is zero for non-successful attacks, and otherwise, the payoff is equal to the surrogate model's test accuracy. At the start of the game, both players choose their strategy from the payoff matrix. We observe that the defender maximizes their gain if they minimize the payoff, whereas the attacker wants to maximize the payoff. A *Nash equilibrium* is found when neither player gains from changing their chosen parameters. Optimal parameters for both players can be derived as follows.

$$(d^*, a^*) = (d_i, a_j) = \arg\min_i(\arg\max_j V[i, j]) \qquad (4)$$

Using the Nash equilibrium to present our results, we demonstrate that successful watermark removal attacks exist due to the watermarking scheme's vulnerability rather than a wrong choice of parameters.

VI. Experiments

In this section, we present the results of our experiments. We describe our experimental setup, a methodology for splitting data between the attacker and defender, and the model architectures. Then, we report measured quantities of the attacks and schemes, such as their runtimes or the embedding loss.

We analyze the robustness of each watermarking scheme against (i) all attacks, (ii) categories of attacks, and (iii) individual attacks. The first experiment validates whether a scheme is robust if the attacker knows which scheme the defender has chosen (but not its parameters). The second experiment analyzes which attack categories are most effective against each watermarking scheme. The third experiment focuses on finding *dominant* attacks, i.e., successful removal attacks that remove any watermark. Our results show that none of the single attacks on their own removes all watermarks. Still, we can find *combined* attacks that are dominant. We cannot depict all evaluation results in this paper. Hence, we will make our results publicly available via an interactive graph that shows the Nash equilibrium for a set of attacks against a set of watermarking schemes[3].

A. Setup

We implement all watermark schemes and removal attacks in our novel Watermark-Robustness-Toolbox (WRT) with PyTorch [51] running as its backend. WRT will be made available as open-source code, which allows independently verifying our empirical results. All reported runtimes in this paper were obtained using (single) Tesla P100 GPUs.

B. Datasets

We embed watermarks into source models trained on the image classification datasets CIFAR-10 [27] and ImageNet [28]. The Open Images [49] dataset is used as a transfer dataset (see Section IV-B). Our method of splitting the dataset between the attacker and defender differs depending on the attack's category. For model modification attacks, the attacker has access to a third of the dataset and the defender can access the remaining two thirds. Model extraction attacks require more data to achieve a high test accuracy, hence the attacker and defender have access to the entire training dataset. We refer

[3]https://crysp.uwaterloo.ca/research/mlsec/wrt

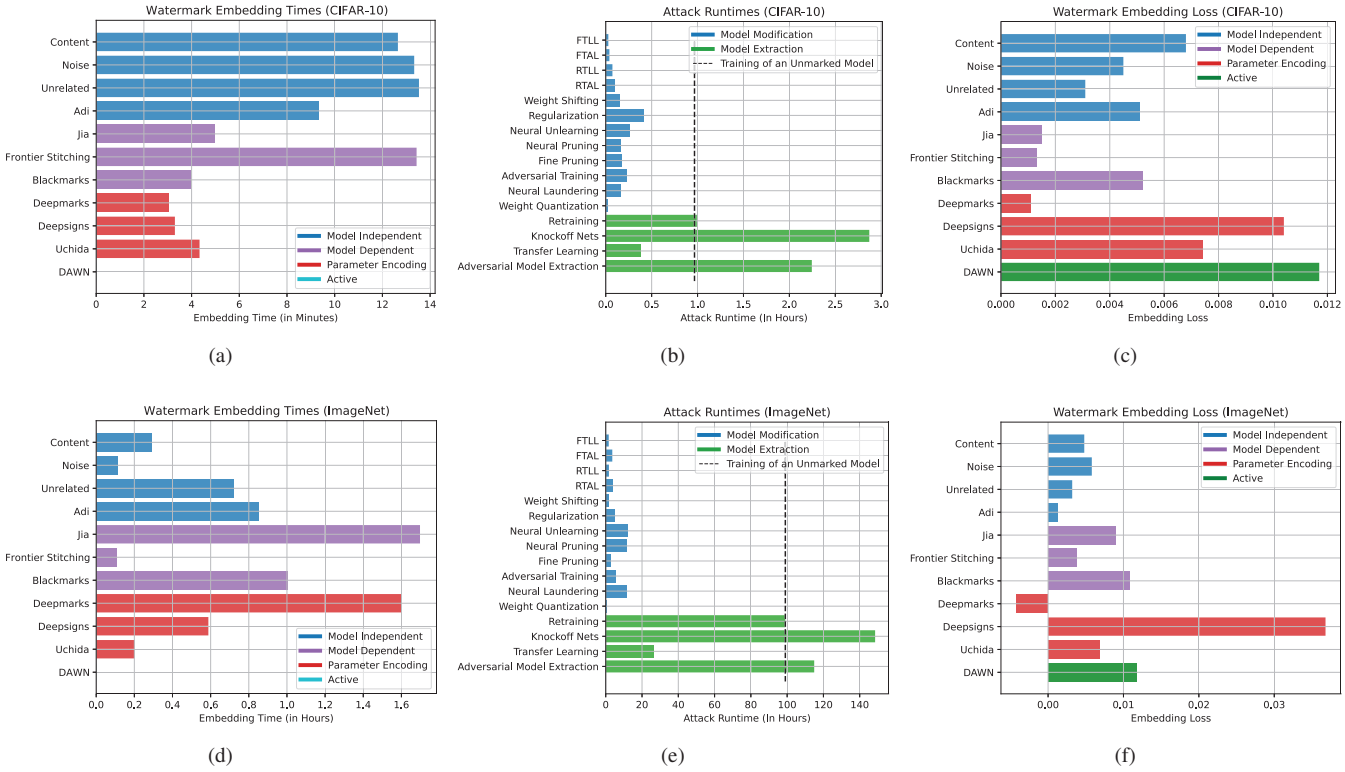

Fig. 2: The measured runtimes for embedding and attacking a watermark and the embedding losses for each watermark on CIFAR-10 (top) and ImageNet (bottom). Figures 2a, 2d show the embedding times and Figures 2b, 2e show the removal attack runtime. Figures 2c, 2f show the embedding loss of each watermarking scheme, which is the difference in test accuracy between an unmarked model and the (marked) source model.

to Appendix XII for a description of the datasets and details on our method of splitting the training dataset.

C. Model Architectures

All of our experiments assume that the attacker knows the source model's architecture. For CIFAR-10, we use the wide ResNet 28x10 [52] and for ImageNet the ResNet-50 [53] architectures. We also perform cross-architecture retraining using a DenseNet-121 [54] for CIFAR-10 and ImageNet.

D. Runtimes and Embedding Losses

We report the runtimes for the removal attacks and watermark embeddings. Since the runtimes are influenced by the choice of parameters, the results can only show general trends. We ensured choosing parameters and training configurations that an attacker or defender would also likely choose in practice, such as using early stopping for the embedding. For a detailed description of the chosen parameters and implementation details we refer to Appendixes X and XI. Figures 2a to 2c show results for CIFAR-10 and Figures 2d to 2f for ImageNet. All graphs are shown as horizontal bar charts with the watermarking scheme or removal attack on the y-axis and the runtime or the embedding loss on the x-axis. The coloring indicates the category of a scheme or removal attack.

Embedding Runtimes. Figures 2a and 2d show the embedding runtimes for CIFAR-10 and ImageNet. We refer to the *training time* as the time it takes to train an unmarked model from scratch. This training time serves as a point of reference to assess the practicality of removal attacks and watermarking schemes. For CIFAR-10 and ImageNet we observe a training time of 1h and 100h, respectively.

On CIFAR-10, model independent schemes have the highest embedding time of about 20% of the training time, whereas parameter encoding schemes have the lowest embedding times and require only about 9% of the training time. We do not consider the runtime for the active scheme DAWN but point out that deploying DAWN incurs computational costs for each inference. On ImageNet, we observe that schemes such as Jia and Deepmarks require considerably more time than on CIFAR-10, whereas model independent schemes are relatively fast to embed. The longest embedding time has Jia with more than 1.6% of the training time. These embedding times are low compared to the training times for both datasets, and we conclude that all surveyed schemes are efficient.

Attack Runtimes. Figures 2b and 2e show the attack runtimes for CIFAR-10 and ImageNet. Input Preprocessing attacks are not shown, because they run only during inference. We observe that the runtimes of all attacks are proportionally

similar on CIFAR-10 and ImageNet. On both datasets, model extraction attacks require significantly longer than model modification attacks. Transfer learning is an exception for a model extraction attack that is relatively fast as it requires about 40% of the training time on CIFAR-10 and roughly 25% of the training time on ImageNet. Knockoff is the slowest attack which takes considerably longer than retraining due to the larger size of the training dataset.

Embedding Losses. Figures 2c and 2f show the embedding loss for each scheme, which is the drop in test accuracy due to embedding the watermark into the source model (see Section V-A). Embedding losses for CIFAR-10 and ImageNet are about one percentage point, with the exception of Deepsigns on ImageNet, which has an embedding loss of more than three percentage points. The parameter encoding scheme Deepmarks incurs the lowest embedding loss on both datasets.

E. Robustness of Watermarking Schemes

In this section, we analyze the robustness of each watermarking scheme against all attacks. This means that the defender can choose from a set of parameters for a single watermarking scheme, whereas an attacker can choose from all parameters for all removal attacks. The goal of this analysis is to evaluate whether any watermarking scheme can be considered robust against an adaptive adversary. We assume that the attacker knows the watermarking scheme chosen by the defender but not its parameters.

Robustness. The results are illustrated in Figures 3b and 3e in the form of a scatter plot. The x-axis shows the stealing loss, which is the drop in test accuracy in the surrogate model compared to the source model, and the y-axis shows the rescaled watermark accuracy (see Section V-A). A watermark accuracy lower than $\theta' = 0.5$ means that the watermark has been removed. We highlight θ' by a dashed line in the graph. We draw the *Pareto frontier*, which is the set of watermarking schemes with a watermark accuracy or stealing loss so that no other watermarking scheme improves upon both metrics. Jia, Content, and Deepmarks are members of the Pareto frontier for CIFAR-10 and only Jia for ImageNet.

We observe that none of the watermarking schemes is robust. For CIFAR-10, the marked source models can be stolen with a stealing loss of less than one percentage point, i.e., without a considerable loss of utility. For ImageNet, we observe that removal attacks incur a higher stealing loss overall. Jia has the highest stealing loss of three percentage points, whereas the remaining watermarking schemes have a stealing loss of at most two percentage points. We designed a set of adaptive attacks against a subset of watermarking schemes and feature their results separately as following. We refer to Appendix XI for a detailed description of all attacks.

- **Smooth Retraining**: The smooth retraining attack is adapted to the active watermarking scheme DAWN. The idea is to query DAWN multiple times with the same image, using a different affine transformation (e.g., cropping, horizontal flipping) for each query. The label for each image is the mean over all received labels for each image. Smooth retraining is the only attack that removes DAWN on CIFAR-10.
- **Feature Permutation**: Hidden layer neurons are permutation invariant, meaning that we can apply a random permutation on the features without losing any utility of the model. We observe that Deepsigns is the only scheme that is not robust against feature permutation attacks.
- **Weight Shifting**: Weight shifting perturbs the filter weights of each convolutional layer by the negative mean over all its filters, adds a small amount of noise, and fine-tunes the model. We observe that weight shifting is the only model modification attack that removes Uchida on CIFAR-10 and ImageNet.

Fastest Attacks. Figures 3c and 3f show the fastest attacks that successfully remove a watermark. On CIFAR-10, we observe that some schemes such as Deepsigns, Blackmarks, and Adi can be removed with a negligible runtime, whereas Jia and Unrelated require the highest runtime. On ImageNet, we observe that the removal of the watermarks from Unrelated and Jia requires the highest runtime, whereas parameter encoding schemes can be removed in the shortest amount of time. For both datasets, we observe that the fastest attacks depend on the watermarking scheme, i.e., there is no single fastest attack or attack category against all watermarking schemes.

Dataset Availability. We stated that the dataset available to a model extraction attack is larger than for model modification attacks. We ablate over the amount of data available to the attacker to achieve a given test accuracy. This is relevant to discuss the practicality of model extraction attacks because the attacker wants to minimize both (i) the training time and (ii) the amount of data required to perform an attack.

Figures 3a and 3d show the amount of unlabeled data in relation to the surrogate model's test accuracy for CIFAR-10 and ImageNet. The attacker trains their surrogate model on data labeled by source models with a test accuracy of 94.20% on CIFAR-10 and 75.48% on ImageNet. On CIFAR-10, we observe that transfer learning achieves a significantly higher test accuracy than retraining from scratch using the same amount of data. Retraining requires at least about 20k samples to perform a successful attack, whereas transfer learning needs only about 5k samples. On ImageNet, the difference between retraining and transfer learning goes to zero when more than 250k samples are available to the attacker. Performing a successful removal attack requires at least 500k samples. While transfer learning still requires the same amount of data as retraining from scratch, we point out that transfer learning requires significantly less computation time.

F. Robustness against Attack Categories

In the previous section, we showed that none of the watermarking schemes is robust against all attacks. We further analyze the robustness of each watermarking scheme against categories of removal attacks. The defender can choose from the set of parameters for each watermarking scheme, and the attacker can choose from the set of parameters for attacks of only one category. This analysis provides insights into

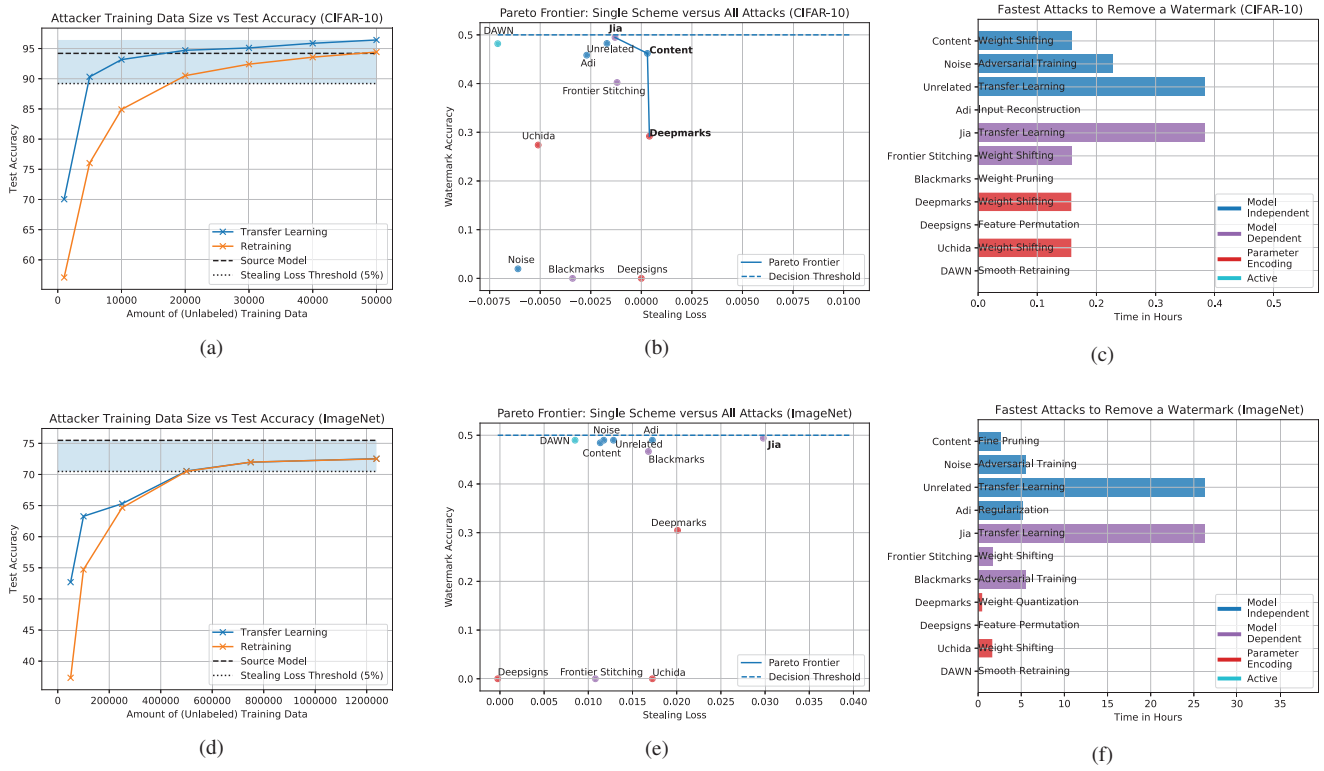

Fig. 3: Figures (a,d) compare the amount of training data required for the transfer learning and retraining attacks to achieve a given test accuracy. Figures (b,e) show the Pareto frontier for all watermarking schemes with respect to the stealing loss (defined in Section V-A) and watermark accuracy of the best attack. A watermark accuracy lower than $\theta' = 0.5$ means that the watermark is not robust. Figures (c, f) show the fastest attack that removes each watermark. For DAWN, the attacker has to obtain white-box access by extracting the source model before using other attacks. For a fair comparison with other schemes, we do not consider this extraction runtime.

the vulnerability of watermarking schemes to certain attack categories. We refer the reader to Table III for a list of all attacks and their categories.

Figure 4 shows a radar plot of our result for CIFAR-10 and ImageNet. The radar plot axis shows the watermark accuracy of each scheme against the best, successful attack from each attack category. A larger covered area of the watermarking scheme in the plot illustrates higher robustness to multiple attack categories. A scheme is robust against the attack category if the watermark accuracy is at least $\theta' = 0.5$ (see Section V-A). We analyze the results for each category.

Input Preprocessing. We observe that input preprocessing attacks often do not remove a watermark on either CIFAR-10 or ImageNet, but these attacks often impact the watermark accuracy. Input smoothing and input reconstruction are effective against Adi and Noise on CIFAR-10, but not on ImageNet. We always apply feature permutation because it does not impact the model's utility and requires negligible computational costs. For this reason, Deepsigns, which is vulnerable to feature permutation, is removed by input preprocessing attacks for both CIFAR-10 and ImageNet. Similarly, DAWN is not robust because it requires extracting a surrogate model prior to running an input preprocessing or model modification attack. We extract a surrogate model for DAWN using smooth retraining, which already removes the watermark.

Model Modification. Model modification attacks are successful at removing all watermarks for CIFAR-10 and ImageNet, except for Jia on ImageNet. Many surveyed watermarking schemes are vulnerable against multiple model modification attacks, whereas other schemes such as Uchida are only vulnerable to our adaptive weight shifting attack. Similar to input preprocessing attacks, we observe that model modification attacks that do not remove the watermark can still significantly lower the watermark accuracy.

Model Extraction. We observe that almost none of the schemes is robust to model extraction attacks on CIFAR-10 and ImageNet. The most effective attack is transfer learning for both CIFAR-10 and ImageNet because it requires a fraction of the training time for an unmarked model, and it removes almost all of the surveyed watermarks. Notable exceptions are Noise and Blackmarks, which are robust against transfer learning on ImageNet, but Noise is not robust against retraining on ImageNet and Blackmarks is not robust against adversarial training. Retraining, distillation, and adversarial training from

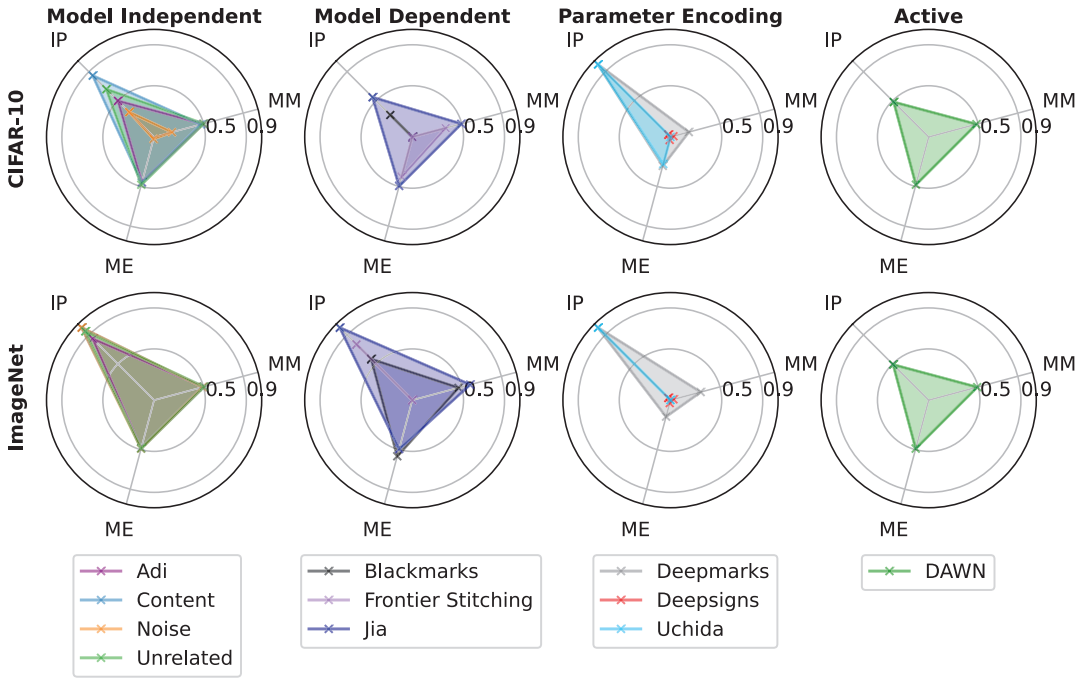

Fig. 4: This figure illustrates the robustness of each surveyed watermarking scheme against categories of attacks for CIFAR-10 (top) and ImageNet (bottom). The axes show the (scaled) watermark accuracy of a scheme against the best attack from each category. A watermark is robust against a category if the watermark accuracy is at least $\theta' = 0.5$. The scheme and attack parameters are chosen using the Nash Equilibrium, and we ignore attacks when their stealing loss exceeds five percentage points. The attack categories are Input Preprocessing (IP), Model Modification (MM), and Model Extraction (ME).

scratch yield similar results as transfer learning, but they require (i) at least as much data and (ii) have a significantly longer runtime. Therefore we do not evaluate distillation and adversarial model extraction on ImageNet if a model is already vulnerable to transfer learning or retraining.

In summary, we conclude that model extraction attacks are the most effective removal attacks against a majority of watermarks. Jia and Blackmarks are robust against retraining, but Jia is not robust against transfer learning, and Blackmarks is not robust against adversarial training. Even when a scheme is robust to retraining with the same architecture, the attacker can obtain a well-trained surrogate model by switching to a different architecture. We believe that transfer learning is more effective at removing some watermarks because the model re-uses low-level features learned from another task. Hence, watermarks encoded into low-level features are less likely to be robust against transfer learning. None of the parameter encoding schemes is robust to transfer learning, also because extraction of such a watermark is not defined for a different model architecture. For example, Uchida defines a secret watermarking key that expects a layer's weights to be in the same shape as the source model's layer used for the embedding. Input preprocessing attacks are often non-successful at removing a watermark, but they can reduce the watermark accuracy. Model modification attacks, especially our novel adaptive attacks, are successful in removing the watermark of a subset of watermarking schemes and require (i) significantly fewer data and (ii) computational resources than model extraction attacks.

G. Attack's Effectiveness.

Table V shows whether a scheme is robust against an attack on CIFAR-10 and ImageNet for a subset of attacks. We make the observations that (i) attacks designed against one category of watermarks are not necessarily effective against watermarks from this category, and (ii) no scheme is robust against all model extraction attacks. Neural Cleanse [35] and Regularization [17] were designed against model independent watermarks, but they often only decrease the watermark accuracy instead of removing the watermark. Jia is robust against retraining, but not against transfer learning suggesting that it is encoded into the low-level features of the source model. Transfer learning does not re-learn these low-level features from scratch, which could explain why transfer learning is more effective than retraining at removing the Jia watermark.

H. Dominant Attacks

This section analyzes whether a *dominant* attack exists that removes all watermarks. The existence of a dominant attack would mean that an attacker does not require knowledge about the scheme used by the defender to remove their watermark.

Attack \ Watermark	Content [16]	Noise [16]	Unrelated [16]	Adi [15]	Jia [24]	FS [25]	BM [23]	Deepmarks [22]	Deepsigns [14]	Uchida [13]	DAWN [26]
INPUT PREPROCESSING											
Input Smoothing [43] (Gaussian Kernel)	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✗/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓
MODEL MODIFICATION											
Regularization [17]	✗/✓	✓/✓	✓/✓	✗/✗	✗/✓	✗/✓	✗/✓	✓/✓	✓/✓	✓/✓	✓/✗
Neural Cleanse [37] (Unlearning)	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓
Feature Permutation (Ours)	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✗/✗	✓/✓	✓/✓
Weight Shifting (Ours)	✗/✓	✓/✓	✗/✓	✗/✓	✓/✓	✓/✗	✗/✓	✗/✗	✗/✗	✗/✗	✓/✓
MODEL EXTRACTION											
Knockoff Nets [40]	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✗/✓	✗/✓	✗/✓	-
Retraining [36]	✗/✗	✗/✗	✗/✗	✗/✗	✓/✓	✗/✓	✗/✗	✗/✗	✗/✗	✗/✗	✓/✗
Smooth Retraining (Ours)	-	-	-	-	-	-	-	-	-	-	✗/✗
Cross-Architecture Retraining	✗/✗	✗/✗	✗/✗	✗/✗	✗/✓	✗/✓	✗/✓	✗/✗	✗/✗	✗/✗	✓/✗
Transfer Learning [39]	✗/✗	✗/✓	✗/✗	✗/✗	✗/✗	✗/✗	✗/✓	✗/✗	✗/✗	✗/✗	✓/✗

TABLE V: A summary of the robustness for each watermarking scheme against selected attacks. A checkmark ('✓') indicates that the scheme is robust, whereas a cross ('✗') indicates that the scheme is *not* robust to this attack. A dash indicates that the attack has not performed against the watermarking scheme (e.g., because it is an adaptive attack designed against a subset of schemes). By two consecutive marks, we indicate the robustness on CIFAR-10 and ImageNet.

The attacker can choose from the set of parameters for a single attack, whereas the defender can choose from the set of parameters for all watermarking schemes. We observe that transfer learning is dominant for CIFAR-10, but there exists no dominant attack for ImageNet.

Creating Dominant Attacks. We now evaluate whether it is possible to find *combined* attacks that are dominant for source models trained on ImageNet. A combined attack performs many attacks in sequence. Our empirical results show that transfer learning combined with label smoothing is a dominant attack that removes all eleven watermarks on CIFAR-10 and ImageNet. The threat of combined attacks to the robustness of watermarking schemes has not yet been explored, and we show that combined attacks can pose a significant threat.

VII. DISCUSSION

In this section, we discuss the practicality of the evaluated removal attacks and argue that they are real-world threats to DNN watermarking. We identify three requirements for the attacker: (1) computational resources, (2) dataset availability, and (3) pre-trained models for transfer learning. Then we present guidelines for designing future watermarking schemes and discuss the implications of our work for future research.

Computational Resources. Related work often restricts the availability of computational resources to the attacker in their threat model [13], [15], [22] and claims robustness against attackers with limited computational resources. We believe that this assumption is not realistic and that a motivated attacker is not limited by computational resources. While it may be the adversary's objective to minimize computational resources, there is no theoretical guarantee that the adversary's learning problem will be a hard instance and require infeasible resources in some security parameters. Quite to the contrary, for the classification problems considered in this paper, the adversary's costs are very feasible. Using shared GPUs in the cloud, the monetary costs are proportional to the attack's runtime. All runtimes in our paper were obtained on (single) Tesla P100 GPUs, which incur a cost of 0.43\$ per on-demand hour of GPU-time[4]. Training a ResNet-50 model from scratch on ImageNet, consisting of 1.28 million images, takes about 100 hours and costs 43\$. Transfer learning a model takes only 23 hours and brings down the costs to about 10\$. There are even more optimized implementations [55] than ours, which achieve lower costs through various optimizations, e.g., by training on multiple GPUs, utilizing TPUs, or choosing more efficient model architectures. We conclude that in absolute terms, the price for computational resources is almost insignificant and is likely not a deterrent for the attacker.

Dataset Availability. Related work often does not put restrictions on the dataset available to the attacker, except for limiting the amount of ground-truth labels. We find that the attacker's dataset significantly influences the effectiveness of the removal attacks. Increasing the amount of (unlabeled) domain data is sufficient to perform successful removal attacks, and predicted labels can substitute ground-truth labels.

We found that using a transfer dataset (labeled data from a different domain) to train a model from scratch, such as in the Knockoff attack [40], does not lead to successful removal attacks. For CIFAR-10, almost all watermarks are retained, and for ImageNet we could not train a surrogate model with high test accuracy. We observe that access to domain data is crucial to perform these attacks.

Availability of Pre-Trained Models. Related work has not used transfer learning to remove watermarks, but transfer learning is a known method for training models in the visual domain [39]. We show that transfer learning is highly effective at removing watermarks; it is computationally efficient, and it can leverage access to less data than other model extraction attacks. Related work has shown that access to larger transfer sets can reduce the amount of domain data required for transfer

[4]https://cloud.google.com/compute/gpus-pricing

learning [56]. Specifically, the authors use models that have been pre-trained on up to 300 million images and show that they can transfer learn this model for ImageNet with a test accuracy of 87.5% using as few as ten examples per class. We argue that it should not be a problem for an attacker to obtain access to a pre-trained model from a different domain in practice. There exist many platforms to share pre-trained models with various model architectures, such as ONNX[5] or Model Zoo[6], without charging the user.

A. Guidelines

In this section, we propose guidelines for evaluating the robustness of watermarking schemes. These guidelines incorporate many of our findings and provide a minimal checklist to claim robustness for a watermarking scheme.

Attacker's Dataset. Our experiments have shown that robustness on CIFAR-10 does not imply robustness on ImageNet and vice versa. In general, we observed that it is more difficult to remove watermarks from models trained on ImageNet than from models trained on CIFAR-10. We believe that is because (i) the model and task are more complex and (ii) attacks have a greater impact on the model's utility (measured by the test accuracy). Our recommendation for image classification models is to experiment on (i) a small dataset, (ii) a dataset with large input image dimensions, and (iii) a dataset with a large number of classes. We use ImageNet to cover the last two requirements within one dataset. Furthermore, we recommend listing the amount of data and ground-truth labels used during the attack for removal attacks.

Decision Threshold. We noticed that a method to derive a watermarking scheme's decision threshold is missing from many papers in related work. Disproving the robustness claim of a scheme requires a method of deriving the decision threshold. This method affects the scheme's usability. For example, for the watermarking scheme Adi, we could theoretically derive the decision threshold because the input images and target labels are drawn randomly. However, Blackmarks requires an empirical method to derive a decision threshold because it relies on adversarial examples for which it is difficult to theoretically quantify the transferability of these examples to unmarked models. Our work proposes a general method to empirically determine this decision threshold, which involves training many unmarked models on CIFAR-10 and ImageNet (hence the usability is limited).

Parameter Ablation. We recommend stating all parameters for a removal attack and watermarking scheme that can be included in an ablation study. In our paper, we manually selected parameters to include in our ablation study. For multiple parameters, the robustness should be evaluated at the Nash equilibrium. This enhances (i) reproducibility of robustness claims and (ii) allows for a fair evaluation of a scheme's robustness and an attack's effectiveness.

Class Accuracies. For some watermarking schemes, such as Content or Jia, we observed that the source model might unlearn a single class during the embedding process. On ImageNet, the test accuracy drops only by about 0.1% when the model unlearns a single class, but we argue that in such cases, the impact of the watermark is greater than the drop in overall test accuracy is suggesting. We recommend to evaluate the drop in test accuracy for single classes.

Runtime. We suggest that a watermarking scheme or removal attack should show their runtimes for the embedding or removal procedure in relation to retraining a model from scratch. While the runtime of all surveyed watermarking schemes is small, we believe the runtime is still a distinguishing factor for the proposed scheme's practicality.

B. Implications for Future Research

We show with our systematic, empirical study that a well-defined attacker can break all surveyed watermarking schemes. We argue that DNN watermarking robustness needs to be defined and evaluated more rigorously. Many previous works evaluate against a relatively weak attacker that does not adapt their attacks. In other cases, the attacker is limited by their computational resources or the non-availability of other pre-trained models. We present a well-defined attacker model and our Watermark-Robustness-Toolbox[7] is publicly available. Authors of future watermarking schemes can evaluate robustness against the attacker presented in this paper. Our paper does not imply that DNN watermarking is impossible and there exist fingerprinting schemes [57] that show promising results.

VIII. Conclusion

We have proposed taxonomies for DNN watermarking schemes and removal attacks. The taxonomies define four categories of watermarking schemes and three categories of removal attacks. We evaluate eleven watermarking schemes from related work and empirically determine their decision thresholds for the CIFAR-10 and ImageNet datasets. Then, we measured the performance of a large set of removal attacks against all watermarking schemes and ablate over multiple parameters for each scheme and removal attack. We use the Nash equilibrium to evaluate a scheme's robustness against (i) all attacks, (ii) categories of attacks, and (iii) single attacks. Our results show that none of the schemes is robust against all attacks. We break down these results by analyzing each attack category's effectiveness and find that the most effective removal attack category are model extraction attacks, followed by model modification attacks. We show that transfer learning removes all watermarks on CIFAR-10, but there exists no such dominant attack for ImageNet. We create a combined attack composed of (1) transfer learning and (2) label smoothing that removes all eleven watermarks. Finally, we discuss the practicality of the removal attacks, e.g., their monetary costs and the dataset availability of the attacker and propose guidelines for evaluating the robustness of DNN watermarking. We hope that our work will improve future evaluations of DNN watermarking schemes.

[5]https://onnx.ai/

[6]https://modelzoo.co/

[7]https://github.com/dnn-security/Watermark-Robustness-Toolbox

REFERENCES

[1] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," 2015.

[2] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[3] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5265–5274.

[4] J. Zhang, Y. Xie, Q. Wu, and Y. Xia, "Medical image classification using synergic deep learning," *Medical image analysis*, vol. 54, pp. 10–19, 2019.

[5] H. Luo, Y. Yang, B. Tong, F. Wu, and B. Fan, "Traffic sign recognition using a multi-task convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1100–1111, 2017.

[6] G. Press, "Cleaning big data: Most time-consuming, least enjoyable data science task, survey says," 2016 (accessed July 5, 2020). [Online]. Available: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/

[7] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer, "Thieves on sesame street! model extraction of bert-based apis," 2020.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 399–414.

[10] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1345–1362.

[11] N. Carlini, M. Jagielski, and I. Mironov, "Cryptanalytic extraction of neural network models," in *Annual International Cryptology Conference*. Springer, 2020, pp. 189–218.

[12] B. G. Atli, S. Szyller, M. Juuti, S. Marchal, and N. Asokan, "Extraction of complex dnn models: Real threat or boogeyman?" in *International Workshop on Engineering Dependable and Secure Machine Learning Systems*. Springer, 2020, pp. 42–57.

[13] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017, pp. 269–277.

[14] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for ip protection of deep learning models," *arXiv preprint arXiv:1804.00750*, 2018.

[15] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1615–1631.

[16] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 159–172.

[17] M. Shafieinejad, J. Wang, N. Lukas, X. Li, and F. Kerschbaum, "On the robustness of the backdoor-based watermarking in deep neural networks," *arXiv preprint arXiv:1906.07745*, 2019.

[18] X. Liu, F. Li, B. Wen, and Q. Li, "Removing backdoor-based watermarks in neural networks with limited data," *arXiv preprint arXiv:2008.00407*, 2020.

[19] T. Wang and F. Kerschbaum, "Attacks on digital watermarks for deep neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2622–2626.

[20] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[22] H. Chen, B. D. Rohani, and F. Koushanfar, "Deepmarks: a digital fingerprinting framework for deep neural networks," *arXiv preprint arXiv:1804.03648*, 2018.

[23] H. Chen, B. D. Rouhani, and F. Koushanfar, "Blackmarks: Blackbox multibit watermarking for deep neural networks," *arXiv preprint arXiv:1904.00344*, 2019.

[24] H. Jia, C. A. Choquette-Choo, and N. Papernot, "Entangled watermarks as a defense against model extraction," *30th {USENIX} Security Symposium ({USENIX} Security 21) (to appear)*, 2021.

[25] E. Le Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications*, vol. 32, no. 13, pp. 9233–9244, 2020.

[26] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "Dawn: Dynamic adversarial watermarking of neural networks," *arXiv preprint arXiv:1906.00830*, 2019.

[27] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[29] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," *Image*, vol. 2, p. T2.

[30] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "Sok: How robust is deep neural network image classification watermarking? (extended version)," in *IEEE Symposium on Security and Privacy*, 2022.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[33] H. Li, E. Wenger, B. Y. Zhao, and H. Zheng, "Piracy resistant watermarks for deep neural networks," *arXiv preprint arXiv:1910.01226*, 2019.

[34] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[35] W. Aiken, H. Kim, and S. Woo, "Neural network laundering: Removing black-box backdoor watermarks from deep neural networks," *arXiv preprint arXiv:2004.11368*, 2020.

[36] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[37] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[38] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.

[39] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[40] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4954–4963.

[41] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[42] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," *arXiv preprint arXiv:1904.08444*, 2019.

[43] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[44] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

[45] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, "Efficient defenses against adversarial attacks," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 39–49.

[46] W.-A. Lin, Y. Balaji, P. Samangouei, and R. Chellappa, "Invert and defend: Model-based approximate inversion of generative adversarial networks for secure inference," *arXiv preprint arXiv:1911.10291*, 2019.

[47] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low pre-

cision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[49] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[50] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[52] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[54] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[55] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "Dawnbench: An end-to-end deep learning benchmark and competition," *Training*, vol. 100, no. 101, p. 102, 2017.

[56] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big transfer (bit): General visual representation learning," *arXiv preprint arXiv:1912.11370*, vol. 6, no. 2, p. 8, 2019.

[57] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," *International Conference on Learning Representations*, 2021.

[58] Z. Cataltepe, Y. S. Abu-Mostafa, and M. Magdon-Ismail, "No free lunch for early stopping," *Neural computation*, vol. 11, no. 4, pp. 995–1009, 1999.

[59] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[60] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[63] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[64] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[65] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[66] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

IX. Appendix

The Appendix is organized as follows. Section XII describes the datasets used in our experiments. Section X describes all surveyed watermarking schemes and the parameters we used in our ablation study. Section XI describes all surveyed removal attacks including novel attacks such as weight shifting and contains a description of the parameters we used in the ablation study. We refer to our technical report [30] for a survey-style description of the watermarking schemes and removal attacks. A detailed description of each approach can be found in the author's papers. Section IV provides further details on the computation of the decision thresholds (see Section V-A).

X. Watermarking Schemes

In this section, we present the surveyed watermarking schemes and the parameters used for our ablation study. For simplicity, we refer to a watermarking scheme by the first author's name unless it is known under a different name.

A. Model Independent

Adi [15]. We embed the same 100 watermarking keys used by the authors[8]. Images are resized along their shortest side to the dimensions of the training data, followed by center cropping. For ImageNet, we embed using early stopping [58] on the watermarking loss with a patience of five, evaluated at the end of every 200th batch. The watermarking loss is the cross-entropy loss of the model computed on the watermarking key. We ablate over the learning rate $\text{lr} \in \{10^{-3}, 10^{-4}\}$. To speed up the embedding, we repeat the watermarking keys 1000 times for ImageNet and 100 times for CIFAR-10.

Zhang [16]. The authors propose three different schemes, referred to as *Content*, *Noise* and *Unrelated*.

- **Content**: We use a white square embedded at the top left corner of the image. The square's size is $s \in \{32, 128\}$ for ImageNet and $s \in \{8, 16\}$ for CIFAR-10.
- **Noise**: We add the noise across the entire image and clip the resulting values into the range $[0, 1]$. We ablate over the standard deviation $\sigma \in \{0.4, 1.0\}$ for both ImageNet and CIFAR-10. For CIFAR-10, we ablate over the learning rate during the embedding $\text{lr} \in \{10^{-3}, 10^{-4}\}$.
- **Unrelated**: We sample watermarking images from the Omniglot dataset [59] for both CIFAR-10 and ImageNet. We ablate over the learning rate $\text{lr} \in \{10^{-3}, 10^{-4}\}$.

For CIFAR-10, we randomly sample the source-target class pair 'cat' and 'dog' and for ImageNet, we sample 'tiger shark' and 'stingray'. We use early stopping on the watermarking loss during the embedding and repeat the watermarking keys 1000 times for ImageNet and 100 times for CIFAR-10.

B. Model Dependent

Frontier-Stitching [25]. We use FGM [31] to generate adversarial examples and ablate over the perturbation threshold perturbation threshold $0.1 \leq \epsilon \leq 0.25$.

[8]https://github.com/adiyoss/WatermarkNN

Blackmarks [23]. We ablate over the loss term that minimizes the bit error rate between the predicted cluster and the assigned cluster $0.01 \leq \lambda \leq 100$.

Jia [24]. We sample the watermarking key from the training data and use a square as the secret trigger pattern (same as the authors). For CIFAR-10, we compute the source class 4 ('deer') and target class 6 ('frog'). We use SNNL weights $w \in \{0.25, 1, 4\}$ and a rate of $r = 2$, i.e., every second batch consists of watermark data. The trigger has a size of 3×3 pixels and resets values to zero in the image across all three channels. For ImageNet, we compute source class 3 ('tiger shark, Galeocerdo cuvieri') and target class 4 ('hammerhead, hammerhead shark'). We use an SNNL weight $w = 64$ and a ratio of ten during the embedding using a square trigger with 5×5 pixels. We compute the SNNL on a single layer, as mentioned by the authors, due to GPU memory restrictions when computing the SNNL on all layers. When embedding 100 elements with a batch size of 64, we observe the convergence of the SNNL and cross-entropy losses after about 100k images are shown to the source model.

C. Parameter Encoding

Uchida [13]. We embed the Uchida watermark with early stopping on the loss during training and a patience of five, whereby we evaluate the condition at the end of every epoch for CIFAR-10 and after every 200 batches for ImageNet. The target layer has $9\,408$ weights for the ImageNet models and 432 for CIFAR-10 models. For CIFAR-10, we ablate over the constant weight factor of the embedding loss $\lambda \in \{0.1, 1, 10\}$ and for ImageNet, we ablate over $\lambda \in \{1, 10\}$.

DeepMarks[9] [22]. We ablate over the embedding strength $\gamma \in \{0.1, 10\}$ and use the same target layer as in Uchida.

DeepSigns [14]. In the author's paper, clusters are modelled using a Gaussian Mixture Model, whereby each feature cluster c_i is described by a mean μ_i and a standard deviation σ_i. In our experiments, we had difficulties embedding the watermark in ImageNet models using more than $m = 1$ Gaussian distributions because of instabilities during training.

Even after extensive parameter search, we observe that for $m > 1$ (i) the test accuracy drops significantly over time, and (ii) the regularization loss does not converge. The authors do not provide source code, nor did they validate their scheme for ImageNet. We solve the issue for ImageNet by modifying two elements of the embedding procedure.

- **Single Gaussian**: We use $m = 1$ Gaussian and $n = 100$ bits to embed the message on ImageNet. We observe that the regularization loss converges.
- **Alternating Training**: We train on the whole dataset without the regularization loss for two batches. Then, we fine-tune with the embedding loss on samples from the source class for one batch. We observe that this stabilizes training and maintains a high test accuracy.

We can replicate the author's result on CIFAR-10 by using $m = 10$ Gaussian distributions (one for each class) and embedding $n = 10$ bits per Gaussian. On ImageNet, we embed the watermark into a layer with $25\,088$ features and $24\,576$ features for CIFAR-10.

D. Active Schemes

DAWN [26]. We ablate over the expected rate $r \in \{0.01, 0.02\}$ at which a false label is returned.

XI. Watermark Removal Attacks

In this section, we describe the parameters used in our ablation study for all removal attacks surveyed in this paper, sorted by their attack category. We make configuration files that show the parameter ablations for all removal attacks publicly available as part of our Watermark-Robustness-Toolbox (WRT). A summary of the adversary model for each attack (see Section IV) is listed in Table III.

A. Input Preprocessing

Input Reconstruction [46]. uses an autoencoder[10] [60] to compress and reconstruct images before passing them to the surrogate model. We ablate over the size of its bottleneck layer $64 \leq h \leq 512$. We do not perform Input Reconstruction on ImageNet because, to the best of our knowledge, no high-fidelity autoencoder for ImageNet is available.

Input Noising [45]. We ablate over the standard deviation $0.01 \leq \sigma \leq 0.2$ for Gaussian noise with zero mean.

Input Quantization [42]. For a given number of bits b we discretize the input space into 2^b evenly spaced intervals, referred to as *quanta*. We project every value of the input image to the mean of its quantum and ablate over the number of bits $b \in \{3, 4, 5\}$.

Input Smoothing [43]. We use a mean, median, and Gaussian kernel. For the mean and median kernels, we use a filter size of three, and for the Gaussian kernel, we ablate over the standard deviation $0.1 \leq \sigma \leq 0.3$.

Input Flipping. We flip an image along its horizontal axis.

JPEG Compression [44]. We ablate over a parameter $5 \leq q \leq 95$ that controls the quality of the compression.

Feature Squeezing [43]. The quanta values are chosen to be multiples of 0.5^k for some $1 \leq k \leq 6$.

B. Model Modification

Adversarial Training [41]. We inject about 10% of the training dataset's size with adversarial examples generated using Projected Gradient Descent [41] for $\epsilon \in \{0.01, 0.1, 0.25\}$, a step size of 0.01 and a maximum number of 40 iterations. Each adversarial example is repeated twice, and we fine-tune the surrogate model for five epochs.

Feature Permutation. DNNs are invariant to feature permutations, meaning that neurons in a hidden layer can be permuted without affecting the model's functionality. We use (random) feature permutation as an adaptive attack designed specifically against Deepsigns [14], which encodes the message into the activations of hidden layers.

[9]DeepMarks is labeled as a fingerprint by the authors, but since it modifies the model by embedding a message, it is a watermark as per our definition.

[10]https://github.com/foamliu/Autoencoder

Fine-Pruning [38]. We ablate over the sparsity $0.8 \leq \rho \leq 0.95$ and fine-tune for ten epochs on CIFAR-10 and five epochs on ImageNet.

Fine-Tuning [13]. Fine-Tuning as a model stealing attack refers to a set of attacks that first apply a transformation to the model, followed by fine-tuning.

- Fine-Tune All Layers (**FTAL**). All weights are fine-tuned.
- Fine-Tune Last Layer (**FTLL**). All but the last layer's weights are frozen while the model is fine-tuned.
- Retrain All Layers (**RTAL**). The last layer's weights are re-initialized, and all weights are fine-tuned.
- Retrain Last Layer (**RTLL**). The last layer's weights are re-initialized, and only that layer's weights are fine-tuned.

RTAL and RTLL use predicted labels, whereas FTAL and FTLL use ground-truth labels (otherwise, gradients are zero).

Label Smoothing [48]. We use a weight of $\epsilon = 0.3$ for the weighted sum between the prediction and a uniform vector.

Regularization [17]. We L2-regularize for five epochs on CIFAR-10 and one epoch on ImageNet using a weight decay of 0.1 (two orders of magnitudes higher than during training).

Neural Cleanse [37]. We implement both *unlearning* and *pruning* methods proposed by the authors and ablate over the learning rate $10^{-3} \leq \alpha \leq 10^{-2}$ for unlearning and the sparsity $0.8 \leq \rho \leq 0.99$ for pruning.

Neural Laundering [35]. We ablate over the activation threshold to prune convolutional layer neurons $0.03 \leq c \leq 3$ and the learning rate for fine-tuning $10^{-4} \leq \alpha \leq 10^{-2}$.

Weight Pruning [20]. We ablate over the sparsity $0.1 \leq \rho \leq 0.95$ for the trainable weights of each layer.

Weight Shifting. Weight shifting is a novel, adapted attack against parameter encoding watermarking schemes. The idea is to apply a small perturbation to all filters of each convolutional layer in the network, followed by fine-tuning the model to regain the loss in test accuracy. We design weight shifting as an efficient and effective model stealing attack specifically against Uchida [13] and Deepmarks [22].

We explain the attack's idea at the example of Uchida, but a similar intuition holds for Deepmarks where the extraction is highly similar. Let $W \in \mathbb{R}^{n \times c \times w \times}$ be the convolutional filters of a target layer, where n is the number of filters, c are the number of channels, and w, h are the width and height of each filter. A weakness of Uchida exploited by weight shifting is that the attacker knows that if all convolutional filters were inverted, i.e. $W'_i = -W_i$, then the watermark accuracy would be zero. We cannot directly invert all filters, as the model experiences a significant drop in test accuracy. Hence, we construct a 'softer' version of the attack that only moves each filter in the direction of the inverse mean multiplied by some constant weight parameter $\lambda_1 \in \mathbb{R}$. We additionally add small random Gaussian noise to each filter to encourage the network to find slightly different filters in the fine-tuning phase.

Our attack can be formalized by the function $S(W; \lambda_1, \lambda_2)$, which takes as input a set of filters W and outputs a shifted set of filters W'. The parameter λ_1, λ_2 trade off the attack's efficiency with its effectiveness. Let A be a random normal matrix of the same shape as each filter W_i with a variance equivalent to the variance over all filters for a convolutional layer and a mean of zero. Shifted weights for each convolutional layer can be computed by applying the following function.

$$\mathrm{S}(W; \lambda_1, \lambda_2)_i = W_i - \frac{\lambda_1}{n} \sum_{j=1..n} W_j - \lambda_2 A \tag{5}$$

In our experiments, we use $\lambda_1 = 1.5, \lambda_2 = 1.0$ for CIFAR-10 and $\lambda_1 = 1.3, \lambda_2 = 0$ for Imagenet. We fine-tune the model for ten epochs on CIFAR-10 and for five epochs on ImageNet.

Weight Quantization [47]. We ablate over the bit-size $b \in \{4, 5\}$ (i.e., there are 2^b discrete states) for CIFAR-10 and ImageNet and fine-tune the model for one epoch.

C. Model Extraction

Retraining [36]. We use the same parameters for the surrogate model that were used to train the source model.

Smooth Retraining. Smooth Retraining trains a surrogate model on smoothed labels obtained from querying the source model for multiple variations of the same image. For each query, a random, affine transformation (e.g., random cropping) is applied to the image, and the mean of all received labels is computed as the final label. We design smooth retraining as an adaptive attack against the active watermarking scheme DAWN. The intuition is that if DAWN responds with a false label for one image, variations of the same image have a high probability of receiving the label predicted by the source model. In our experiments, we use $n = 3$ queries.

Knockoff Nets [40]. We implement the random selection approach on the Open Images [49] dataset.

Transfer Learning [39]. Transfer Learning is an established method from related work, where a pre-trained model from a different domain is fine-tuned for a new domain. We propose using transfer learning as a novel method to remove DNN watermarks. We use a pre-trained ResNet-101 model[11] for Open Images (v2) [49] that was published by Google in 2017. The model defines an output layer with 5k output classes, which we replace by a layer with ten output classes for CIFAR-10 and 1k output classes for ImageNet. We transfer-learn the model using stochastic gradient descent (SGD) and freeze all but the last layer for the first 300 batches. We proceed by training the entire model for five epochs and reduce the learning rate by a factor of ten in epochs three and four.

Adversarial Training (from scratch) [41]. This method is equivalent to adversarial training described earlier, except that the attacker trains the surrogate model from scratch.

XII. Datasets

We now describe the datasets used in our experiments.

- **CIFAR-10** [27] contains 50k training images and 10k testing images from 10 classes. All images have a resolution of 32×32 pixels.
- **ImageNet** [28] contains 1.28 million training images and 150k testing images from 1k classes. We resize and center crop all images to 224×224 pixels.

[11] https://storage.googleapis.com/openimages/2017_07/oidv2-resnet_v1_101.ckpt.tar.gz

- **Open Images** [49] defines 19.794 classes and contains in total 8.85 million training images, out of which we use a subset of 1.7 million images due to storage constraints on our machines. Images can be labeled by multiple classes. We resize and center-crop all images to 224×224 pixels.

All source models are trained on either CIFAR-10 or ImageNet. The Open Images dataset is only used in the transfer learning attack. We use standard training procedures and data augmentation, such as horizontal flipping, to train models for CIFAR-10 and ImageNet from scratch. On CIFAR-10 and ImageNet, the source models achieve a test accuracy of 94.20% and 75.48% respectively.

A. Dataset Splitting

We split the whole training dataset into thirds and assign two-thirds to the defender for embedding the watermark. For the attacker's training data, we recall from Section IV-B that we distinguish between the availability of the following three datasets to the attacker.

1) **Labeled**: Data from the same distribution where a subset of at most a third of the data is labeled.
2) **Domain**: Unlabeled data from the same distribution.
3) **Transfer**: Labeled data from a different distribution.

In the first two cases, we assign the remaining third of the training dataset to the attacker. We make an exception for model extraction attacks, where the attacker has access to the whole training dataset without labels. Such an exception is necessary because model extraction attacks require a substantial amount of data to output well-trained surrogate models. We underpin this argument by an ablation study in Section VI-E. Otherwise, if the attacker is given domain data, we replace all labels with the predictions of the source model.

XIII. Estimating the Decision Threshold

For model independent, model dependent and active watermarking schemes, we use 20 publicly available, pre-trained models from the torchvision[12] package that do not necessarily share the source model's architecture (ResNet-50). We use the following model architectures.

ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152 [61], Wide ResNet-50, Wide ResNet-101 [52], VGG11, VGG13, VGG16, VGG19 [62], SqueezeNet [63], DenseNet-121, DenseNet-161 [54], GoogleNet [64], Alexnet, Alexnet-50 [65], InceptionNet [48], MobileNetV2 [66]

[12]https://pytorch.org/vision/stable/models.html