DAWN: Dynamic Adversarial Watermarking of Neural Networks

Sebastian Szyller
Aalto University
Finland
sebastian.szyller@aalto.fi

Buse Gul Atli
Aalto University
Finland
buse.atli@aalto.fi

Samuel Marchal
Aalto University & F-Secure Corporation
Finland
samuel.marchal@aalto.fi

N. Asokan
University of Waterloo
Canada
asokan@acm.org

ABSTRACT

Training machine learning (ML) models is expensive in terms of computational power, amounts of labeled data and human expertise. Thus, ML models constitute intellectual property (IP) and business value for their owners. Embedding digital watermarks during model training allows a model owner to later identify their models in case of theft or misuse. However, model functionality can also be stolen via *model extraction*, where an adversary trains a *surrogate model* using results returned from a prediction API of the original model. Recent work has shown that model extraction is a realistic threat. Existing watermarking schemes are ineffective against IP theft via model extraction since it is the adversary who trains the surrogate model. In this paper, we introduce DAWN (Dynamic Adversarial Watermarking of Neural Networks), the first approach to use watermarking to deter model extraction IP theft. Unlike prior watermarking schemes, DAWN does not impose changes to the training process but it operates at the prediction API of the protected model, by dynamically changing the responses for a small subset of queries (e.g., <0.5%) from API clients. This set is a watermark that will be embedded in case a client uses its queries to train a surrogate model. We show that DAWN is resilient against two state-of-the-art model extraction attacks, effectively watermarking all extracted surrogate models, allowing model owners to reliably demonstrate ownership (with confidence $>1 - 2^{-64}$), incurring negligible loss of prediction accuracy (0.03-0.5%).

KEYWORDS

Deep Neural Network, Watermarking, Model stealing, IP protection

1 INTRODUCTION

Recent progress in machine learning (ML) has led to a dramatic surge in the use of ML models for a wide variety of applications. Major enterprises like Google, Apple, and Facebook have already deployed ML models in their products [41]. ML-related businesses are expected to generate trillions of dollars in revenue in the near future [14]. The process of collecting training data and training ML models is the basis of the business advantage of model owners. Hence, protecting the intellectual property (IP) embodied in ML models is necessary.

One approach for IP protection of ML models is *watermarking*. Recent work [1, 31, 48] has shown how *digital watermarks* can be embedded into deep neural network models (DNNs) during training. Watermarks consist of a set of inputs, the *trigger set*, with incorrectly assigned labels. A legitimate model owner can use the trigger set, along with a large training set with correct labels, to train a watermarked model and distribute it to his customers. If he later encounters a model he suspects to be a copy of his own, he can demonstrate ownership by using the trigger set as inputs to the suspected model. These watermarking schemes allow legitimate model owners to detect theft or misuse of their models.

Instead of distributing ML models to customers, an increasingly popular alternative business paradigm is to allow customers to use models via *prediction APIs*. But one can mount a *model extraction* [42] attack via such APIs by sending a sequence of API queries with different inputs and using the resulting predictions to train a *surrogate model* with similar functionality as the queried model. Model extraction attacks are effective even against complex DNN models [21, 33], and are difficult to prevent [21]. Existing watermarking techniques, which rely on model owners to embed watermarks during training, are ineffective against model extraction since it is the adversary who trains the surrogate model.

In this paper we introduce DAWN (Dynamic Adversarial Watermarking of Neural Networks), a new watermarking approach intended to deter IP theft via model extraction. DAWN is designed to be deployed within the prediction API of a model. It dynamically watermarks a tiny fraction of queries from a client by changing the prediction responses for them. The watermarked queries serve as the trigger set if an adversarial client trains a surrogate model using the responses to its queries. The model owner can use the trigger set to demonstrate IP ownership of the extracted surrogate model as in prior DNN watermarking solutions [1, 48]. DAWN differs from them in that it is the adversary (model thief), rather than the defender (original owner) who trains the watermarked model. This raises two new challenges: (1) defenders must choose trigger sets from among queries sent by clients and cannot choose optimal trigger sets from the whole input space; (2) adversaries can select the training data or manipulate the training process to resist the embedding of watermarks. DAWN addresses both these challenges.

DAWN watermarks are client-specific: DAWN not only infers whether a given model is a surrogate but, in case of model extraction, also identifies the client whose queries were used to train the surrogate. DAWN is parametrized so that changed predictions needed for watermarking are sufficiently rare as to not degrade the utility of the original model for legitimate API clients.

We make the following contributions:

- present DAWN, the first approach for dynamic, selective watermarking for DNN models at their prediction APIs for deterring IP theft via model extraction (Sect. 4),
- empirically assess it (Sect. 5) using several DNN models and datasets showing that DAWN is robust to adversarial manipulations and resilient to evasion (Sect. 6), and
- show that DAWN is resistant to two state-of-the-art extraction attacks, reliably demonstrating ownership (with confidence $>1 - 2^{-64}$) with negligible impact on model utility (0.03-0.5% decrease in accuracy) (Sect. 7).

2 BACKGROUND

2.1 Deep Neural Network

A DNN is a function $F : \mathbb{R}^n \to \mathbb{R}^m$, where n is the number of input features and $m \geq 2$ is the number of output classes in a classification task. $F(x)$ is a vector of length m containing probabilities p_j that x belongs to each class $c_j \in C$ for $j \in \{1, m\}$. The predicted class, $\hat{F}(x)$, is obtained by applying the *argmax* function: $\hat{F}(x) = argmax(F(x)) = c$. F is trained such that $\hat{F}(x)$ approximates a perfect oracle function $O_f : \mathbb{R}^n \to C$ which gives the true class c for any sample $x \in \mathbb{R}^n$. If F is successfully trained, $\hat{F} \sim O_f$, and the accuracy of F is close to 1: $Acc(F) = 1 - \epsilon$ where ϵ is the irreducible error.

2.2 Model Extraction Attacks

In model extraction [9, 21, 33, 34, 42], an adversary $\mathcal{A}$ wants to "steal" a DNN model $F_{\mathcal{V}}$ of a victim $\mathcal{V}$ by making a series of prediction requests U to $F_{\mathcal{V}}$ and obtaining predictions $F_{\mathcal{V}}(U)$. U and $F_{\mathcal{V}}(U)$ are used by $\mathcal{A}$ to train a surrogate model $F_{\mathcal{A}}$. $\mathcal{A}$'s goal is to have $Acc(F_{\mathcal{A}})$ as close as possible to $Acc(F_{\mathcal{V}})$. All model extraction attacks [9, 21, 33, 34, 42] operate in a *black-box* setting: $\mathcal{A}$ has access to a prediction API, $\mathcal{A}$ uses the set $< U, F_{\mathcal{V}}(U) >$ to iteratively refine the accuracy of $F_{\mathcal{A}}$. Depending on the adversary model, $\mathcal{A}$'s capabilities can be divided into three categories: *model knowledge*, *data access*, and *querying strategy*.

Model knowledge. $\mathcal{A}$ does not know the exact architecture of $F_{\mathcal{V}}$ or the hyperparameters or the training process. However, given the purpose of the API (e.g., image recognition) and expected complexity of the task, $\mathcal{A}$ may attempt to guess the architecture of the model [21, 34]. On the other hand, if $F_{\mathcal{V}}$ is complex, $\mathcal{A}$ can use a publicly available, high capacity model pre-trained with a very large benchmark datasets [33]. While the above methods focus on DNNs, there are alternatives targeting simpler models: logistic regression, decision trees, shallow neural networks [42].

Data access. $\mathcal{A}$'s main limitation is the lack of access to *natural data* that comes from the *same distribution* as the data used to train $F_{\mathcal{V}}$. $\mathcal{A}$ may use data that comes from the same domain as $\mathcal{V}$'s training data but from a different distribution [9]. If $\mathcal{A}$ does not exactly know the distribution or the domain, it may use widely available natural data [33] to mount the attack. Alternatively, it may use only *synthetic samples* [42] or a mix of a small number of natural samples augmented by synthetic samples [21, 34].

Querying strategy. All model stealing attacks [9, 21, 33, 34, 42] consist of alternating phases of $\mathcal{A}$ querying $F_{\mathcal{V}}$, followed by training the surrogate model $F_{\mathcal{A}}$ using the obtained predictions. $\mathcal{A}$

queries $F_{\mathcal{V}}$ with all its data and then trains the surrogate model [9, 33]. Alternatively, if $\mathcal{A}$ relies primarily on synthetic data [21, 34], it deliberately crafts inputs that would help it train $F_{\mathcal{A}}$.

2.3 Watermarking DNN models

Digital watermarking is a technique used to covertly embed a marker, *the watermark*, in an object (image, audio, etc.) which can be used to demonstrate ownership of the object. Watermarking of DNN models leverages the massive overcapacity of DNNs and their ability to fit data with arbitrary labels [47]. DNNs have a large number of parameters, many of which have little significance for their primary classification task. These parameters can be used to carry additional information beyond what is required for its primary classification task. This property is exploited by backdooring attacks, which consist in training a DNN model that deliberately outputs incorrect predictions for some selected inputs [7, 17].

Watermarking of DNN models is currently based on backdooring attacks [1, 10, 31, 48]. We want to train a DNN model $F : \mathbb{R}^n \to \mathbb{R}^m$ for which the primary task is that $\hat{F}(x) = argmax(F(x)) = c$ approximates an oracle $O_f : \mathbb{R}^n \to C$. Embedding a watermark in F consists of enabling F with a secondary classification task: for a subset of samples $x \in T \subset \mathbb{R}^n$, we want $\hat{F}$ to output incorrect prediction classes as defined by a function $B : T \to \mathbb{R}^m$ such that $\hat{B}(x) \neq O_f(x)$. We call $B(x)$ a *backdoor function* and T a *trigger set*: T triggers the backdoor. F is trained using the trigger set T mislabeled using $\hat{B}(x)$ in addition to a larger set of samples $x \in \mathbb{R}^n \setminus T$ accurately labeled using $O_f(x)$. F is a watermarked DNN model which is expected to approximate the backdoor function $B(x)$ for $x \in T$ and the oracle O_f for $x \in \mathbb{R}^n \setminus T$.

The trigger set T and the outputs of the backdoor function for its elements $\hat{B}(T)$ compose the watermark: $(T, \hat{B}(T))$. Let F' be a DNN model that copies F. The watermark can be used to demonstrate ownership of F'. It only requires F' to expose a prediction API which can be used to query all samples in the trigger set $x \in T$. A sufficient number of predictions $\hat{F}'(x)$ such that $\hat{F}'(x) = \hat{B}(x)$ demonstrates that F' is a copy of the watermarked model F.

3 PROBLEM STATEMENT

3.1 Adversary Model

The adversary $\mathcal{A}$ mounts a model extraction attack against a victim model $F_{\mathcal{V}}$ using queries to its prediction API. $\mathcal{A}$'s goal is *model functionality stealing* [33]: train a surrogate model $F_{\mathcal{A}}$ that performs well on a classification task for which $F_{\mathcal{V}}$ was designed. If $\hat{F}_{\mathcal{V}} \sim O_f$ then $\mathcal{A}$'s goal is that $\hat{F}_{\mathcal{A}} \sim O_f$, which can be considered successful if $Acc(F_{\mathcal{A}}) \sim Acc(F_{\mathcal{V}})$. A secondary goal is to minimize the number of queries to $F_{\mathcal{V}}$ necessary for $\mathcal{A}$ to train $F_{\mathcal{A}}$.

$\mathcal{A}$ has full control over the samples $D_{\mathcal{A}}$ it chooses to query $F_{\mathcal{V}}$ with. These can be natural [33] or synthetic [21, 34, 42]. $\mathcal{A}$ obtains a prediction for each query in the form of probability vectors $F_{\mathcal{V}}(x)$ or single classes $\hat{F}_{\mathcal{V}}(x), \forall x \in D_{\mathcal{A}}$. $\mathcal{A}$ uses queried samples and their predictions to train $F_{\mathcal{A}}$, a DNN. It chooses the DNN model architecture, training hyperparameters and training process. Requiring $F_{\mathcal{A}}$ to be a DNN is justified by the observations in prior work on model extraction attacks [21, 33, 34] that $F_{\mathcal{A}}$ needs to have

equal or larger capacity than F_V in order for model extraction to be successful. DNNs have the greatest capacity among ML models [47].

3.2 Assumptions

We assume that for a given input $x \in D_{\mathcal{A}}$, $\mathcal{A}$ has no a priori expectation regarding the prediction $F_V(x)$. $\mathcal{A}$ treats $y = F_V(x)$ as the ground truth label for $x \in D_{\mathcal{A}}$. $\mathcal{A}$ expects that multiple queries of the same input x must return the same prediction y.

Our focus is on $\mathcal{A}$ who makes $F_{\mathcal{A}}$ available via a prediction API since it has the greatest impact on V's business advantage. We do not consider an $\mathcal{A}$ who keeps $F_{\mathcal{A}}$ for private use. This is similar to media watermarking schemes where access to allegedly stolen media is a pre-requisite for ownership demonstration [35].

3.3 DAWN Goals and Overview

On one hand, model extraction attacks against DNNs have been proven difficult to defend against [21]. On the other hand, existing watermarking techniques [1, 10, 31] are vulnerable to model extraction attacks [48]. To address these limitations, we design a solution to identify and prove the ownership of DNN models stolen through a prediction API.

Our solution, DAWN (Dynamic Adversarial Watermarking of Neural Networks), is an additional component added in front of a model prediction API (Fig. 1). DAWN dynamically embeds a watermark in responses to queries made by a *API client*. This watermark is composed of inputs $x_i \in T$ for which we return incorrect predictions $B(x_i) \neq F_V(x_i)$. $\mathcal{A}$ uses all the responses including these mislabeled samples $(x_i, B(x_i))$ to train $F_{\mathcal{A}}$. $F_{\mathcal{A}}$ will remember those samples as a backdoor [7] that represents the watermark (as in traditional DNN watermarking techniques). If $F_{\mathcal{A}}$ exposes a public prediction API, a judge $\mathcal{J}$ can run a verification process (*verify*), which confirms $F_{\mathcal{A}}$ is a surrogate of F_V. *Verify* checks that for sufficient number of inputs $x_i \in T$, we have $\hat{F}_{\mathcal{A}}(x_i) = \hat{B}(x_i) \neq \hat{F}_V(x_i)$. DAWN embeds a watermark into a subset of queries it receives so that any $F_{\mathcal{A}}$ trained using these responses will retain the watermark.

3.4 System requirements

We define the following requirements for the watermark that DAWN embeds in $F_{\mathcal{A}}$ during an extraction attack. **W1-W3** were introduced in [1] while **W4** is a new requirement specific to DAWN.

W1 Unremovability: $\mathcal{A}$ is unable to remove the watermark from $F_{\mathcal{A}}$ without significantly decreasing its accuracy, rendering it "unusable". If F_V is free of the watermark, then $Acc(F_{\mathcal{A}}) \ll Acc(F_V)$.

W2 Reliability: If *verify* outputs "true" for a watermark $(T, \hat{B}_V(T))$ on a model F', then F' is a surrogate of F_V, with high confidence. On the other hand, if F' is not a surrogate, $\mathcal{A}$ cannot generate a watermark $(T, \hat{B}(T))$ such that *verify* outputs "true" (non-trivial ownership).

W3 Non-ownership piracy: $\mathcal{A}$ cannot produce a watermark for a model that was already watermarked by V, such that it can cast V's ownership into doubt.

W4 Linkability: If *verify* outputs "true" for a model $F_{\mathcal{A}}$, the watermark used for verification $(T, \hat{B}(T))$ can be linked to a specific API client whose queries were used to train $F_{\mathcal{A}}$.
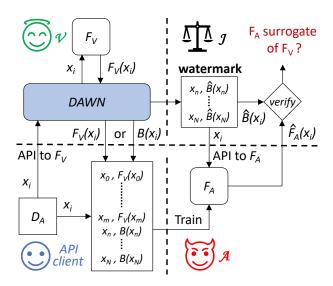

Figure 1: DAWN system overview with four parties: a victim V owning a model F_V, API clients querying the model prediction API, an adversary $\mathcal{A}$ training a surrogate model $F_{\mathcal{A}}$ and a judge $\mathcal{J}$ verifying the surrogacy of $F_{\mathcal{A}}$.

We identify additional requirements **X1-X3**:

X1 Utility: Incorrect predictions returned by DAWN do not significantly degrade the prediction service provided to legitimate API clients: $Acc(\text{DAWN} + F_V) \sim Acc(F_V)$.

X2 Indistinguishability: $\mathcal{A}$ cannot distinguish incorrect predictions $B(x)$ from correct victim model predictions $F_V(x)$.

X3 Collusion resistance: Watermark unremovability (**W1**), linkability (**W4**) and indistinguishability (**X2**) must remain valid even if the extraction attack is distributed among several API clients.

In contrast to traditional DNN watermarking, DAWN does not aim at maximizing the accuracy of the watermarked model $Acc(F_{\mathcal{A}})$ on a primary classification task.

3.5 Relation to other attacks

DAWN is different from prior work where the goal is to (a) degrade model performance (decrease test accuracy Acc_{test} – typical of poisoning attacks [2, 32]), (b) trigger targeted misclassifications (classify a trigger set with high accuracy Acc_{bd} – typical to backdooring [28]) or (c) embed a watermark while preserving high model performance (reach high Acc_{bd} and Acc_{test} – typical of DNN watermarking [1, 31]). In contrast to backdooring, DAWN cannot inject arbitrary samples in $D_{\mathcal{A}}$ but it can modify the label of $D_{\mathcal{A}}$ samples to any incorrect prediction $c \neq \hat{F}_V(x)$. In contrast to traditional DNN watermarking, V neither controls the training of $F_{\mathcal{A}}$ nor can it choose the trigger set from the whole input space $\mathbb{R}^n$: V is limited to the set of samples $D_{\mathcal{A}}$ submitted by $\mathcal{A}$. Table 1 summarizes these differences.

Table 1: Adversarial watermarking (DAWN) capabilities and goals compared to (a) poisoning attacks, (b) backdoor attacks and (c) prior DNN watermarking.

| | Capabilities | | | Goal | |
	Modify labels	Inject in $D_{\mathcal{A}}$	Control training	Acc_{test}	Acc_{bd}
Poisoning	Yes	Yes	No	Low	–
Backdoor	Yes	Yes	Yes / No	–	High
Watermarking	Yes	Yes	Yes	High	High
DAWN	Yes	**No**	**No**	–	High

4 DYNAMIC ADVERSARIAL WATERMARKS

We first present the method for generating and embedding an adversarial watermark. Then we describe the process for proving ownership of a model using the watermark.

4.1 Watermark generation

We define *watermarking an input x* as returning an incorrect prediction $B_{\mathcal{V}}(x)$ instead of the correct prediction $F_{\mathcal{V}}(x)$. The collection of all watermarked inputs composes the trigger set $T_{\mathcal{A}}$ that will be a backdoor to any $F_{\mathcal{A}}$ trained using responses from $F_{\mathcal{V}}$ including $T_{\mathcal{A}}$. Consequently, inputs $x \in T_{\mathcal{A}}$ and their corresponding prediction classes $\hat{B}_{\mathcal{V}}(x)$ compose the watermark to the surrogate model $(T_{\mathcal{A}}, \hat{B}_{\mathcal{V}}(T_{\mathcal{A}}))$. We define two functions:

- $W_{\mathcal{V}}(x)$: should the response to x be watermarked?
- $B_{\mathcal{V}}(x)$: what is the (backdoored watermark) response?

$\mathcal{A}$ must not be able to predict $W_{\mathcal{V}}(x)$ or distinguish between $B_{\mathcal{V}}(x)$ and $F_{\mathcal{V}}(x)$. The same query, regardless of the API client, must always get the same output. Both functions must be *deterministic* random functions specific to $F_{\mathcal{V}}$ to fulfill these properties.

We use the result of a keyed cryptographic hash function as a source for randomness. We compute $\text{HMAC}(K_w, x)$ using SHA-256, where K_w is a model-specific secret key generated by DAWN and x is an input to $F_{\mathcal{V}}$. If x is a matrix of dimension $d > 1$, it is flattened to a 1-dimensional vector. The result of the hash is split in two parts $\text{HMAC}(K_w, x)[0, 127]$ and $\text{HMAC}(K_w, x)[128, 255]$, respectively used in $W_{\mathcal{V}}$ and $B_{\mathcal{V}}$. These numbers are independent and provide a sufficient source for randomness for each function.

4.1.1 Watermarking decision. $W_{\mathcal{V}}(x)$ is a boolean function. We define r_w as the fraction of inputs to be watermarked out of N inputs submitted by an API client. r_w will define the size of the trigger set $|T_{\mathcal{A}}| = \lfloor r_w \times N \rfloor$. Then:

$$W_{\mathcal{V}}(x) = \begin{cases} 1, & \text{if } \text{HMAC}(K_w, x)[0, 127] < r_w \times 2^{128}. \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The expectation that $W_{\mathcal{V}}$ returns 1 and thus to watermark a sample is uniformly equal to r_w. It is worth noting that DAWN does not differentiate adversaries from benign API clients. Consequently, any API client obtains a rate r_w of incorrect predictions. r_w must be defined to meet a trade-off. A large r_w increases the reliability of ownership demonstration and prevents trivial ownership demonstration **W2** as later discussed in Sect. 4.3. A small

r_w maximizes utility **X1** by minimizing the number of incorrect predictions returned to benign API clients.

4.1.2 Backdoor function. We implement the backdoor function $B_{\mathcal{V}}(x)$ as a function of $F_{\mathcal{V}}(x)$. Our motivations are two-fold. First, this allows for deploying DAWN to protect any model $F_{\mathcal{V}}$ without the need for redefining $B_{\mathcal{V}}$. Second, it makes $B_{\mathcal{V}}(x)$ consistent with correct predictions $F_{\mathcal{V}}(x)$. We define $B_{\mathcal{V}}(x) = \pi(K_{\pi}, F_{\mathcal{V}}(x))$ where $\pi : \mathbb{R}^m \to \mathbb{R}^m$ is a keyed pseudo-random permutation function with secret key K_{π}. Even if an adversary uncovers values $B_{\mathcal{V}}(x)$ for a large number of inputs, it will not be able to infer the function $B_{\mathcal{V}}$. This prevents an adversary from recovering $F_{\mathcal{V}}(x)$ from $B_{\mathcal{V}}(x)$ in case it knows if an input is backdoored.

$\pi(K_{\pi}, F_{\mathcal{V}}(x))$ does not need to permute all m positions of $F_{\mathcal{V}}(x)$ but only those with highest probabilities for the purpose of backdooring. A large number of classes typically have a 0 probability value when m is large. Considering that the number of positions to permute is small, we use the Fisher-Yates shuffle algorithm [13] to implement π. We use $K_{\pi} = \text{HMAC}(K_w, x)[128, 255]$ as the key that determines the permutations performed during the Fisher-Yates shuffle algorithm. A 128-bits key allows for list permutation of up to 34 positions (34 prediction probabilities) in a secure manner.

4.1.3 Indistinguishability. Outputs $B_{\mathcal{V}}(x)$ must be indistinguishable from $F_{\mathcal{V}}(x)$ **X2**. This requirement is partially addressed by our assumption that $\mathcal{A}$ has no expectation regarding predictions obtained from $F_{\mathcal{V}}$ (Sect. 3.2). Nevertheless, our watermarking function $W_{\mathcal{V}}$ is configured by a hash of the input x. A subtle modification δ to x produces a different hash and consequently, a different result $W_{\mathcal{V}}(x) \neq W_{\mathcal{V}}(x + \delta)$. If $\mathcal{A}$ receives different predictions for x and $x + \delta$ for a small δ, it can discard both x and $x + \delta$ from its training set to avoid the watermark.

Therefore we assume that $\mathcal{A}$ expects two similar inputs x and $x + \delta$ to have similar predictions $F_{\mathcal{V}}(x) \sim F_{\mathcal{V}}(x + \delta)$ when δ is small. To enhance indistinguishability, the decision of $W_{\mathcal{V}}$ must be smoothened to return the same result $W_{\mathcal{V}}(x) = W_{\mathcal{V}}(x + \delta)$ and $B_{\mathcal{V}}(x) = B_{\mathcal{V}}(x + \delta)$. This can be achieved using a mapping function $M_{\mathcal{V}} : \mathbb{R}^n \to \mathbb{R}^p$ that projects x to a space where $M_{\mathcal{V}}(x) = M_{\mathcal{V}}(x + \delta)$ for a small δ. $M_{\mathcal{V}}(x)$ is only used as the new input to our hash function such that $\text{HMAC}(K_w, M_{\mathcal{V}}(x)) = \text{HMAC}(K_w, M_{\mathcal{V}}(x + \delta))$. $M_{\mathcal{V}}(x)$ smoothens the decision of $W_{\mathcal{V}}$ and ensures that permutations performed in $B_{\mathcal{V}}$ are the same for similar inputs (π is keyed by the hash result).

$M_{\mathcal{V}}$ could be implemented as an autoencoder which projects inputs x to a latent space $\mathbb{R}^p$ of lower dimension $p < n$, discarding small perturbations [30]. $M_{\mathcal{V}}$ could also be a masking and binning function [8] removing large modifications of a single pixel value (with masking) and small modifications of a large number of pixels (with binning). We chose another solution for $M_{\mathcal{V}}$: use the embedding obtained from a layer in the middle of $F_{\mathcal{V}}$. This is similar to using an autoencoder but it does not require training additional models. The obtained embedding would also be unknown to $\mathcal{A}$ since it does not know $F_{\mathcal{V}}$ (target of extraction attack).

For each input x, we obtain its latent representation $L_{\mathcal{V}}$ based on $F_{\mathcal{V}}$ as $x_L = L_{\mathcal{V}}(F_{\mathcal{V}}, x)$. This ensures that as long as $F_{\mathcal{V}}$'s prediction is resilient to perceptual modifications (e.g. translation, illumination), so is $M_{\mathcal{V}}$. Next, we smoothen x_L by binarizing it based on the median value of each of its features. The median of each feature

value is obtained by querying $F_\mathcal{V}$ using $\mathcal{V}$'s training set, recording corresponding x_L and taking the median. Using 5000 samples and their intermediate representation of length 100, we get 100 feature vectors of length 5000 and thus, 100 median values. We evaluate $M_\mathcal{V}$ in Sect. 6.2.

4.2 Watermark embedding

$\mathcal{A}$ uses the set of inputs $D_\mathcal{A}$ and the corresponding predictions returned by DAWN-protected prediction API of $F_\mathcal{V}$ to train $F_\mathcal{A}$. Approximately $\lfloor r_w \times |D_\mathcal{A}| \rfloor$ samples from $D_\mathcal{A}$ constitute the trigger set $T_\mathcal{A}$ consisting of incorrect predictions $B_\mathcal{V}(x)$. Given that $F_\mathcal{A}$ has enough capacity (large enough number of parameters), it will be able to remember a certain amount of training data having arbitrarily incorrect labels [47]. This phenomenon is called overfitting and it can be prevented using regularization [3]. But it is not effective for DNNs with a large capacity [47]. This is the rationale for the existence of DNN backdoors [28] and for DAWN. We expect our watermark $(T_\mathcal{A}, \hat{B}_\mathcal{V}(T_\mathcal{A}))$ to be embedded as a backdoor in $F_\mathcal{A}$ as a natural effect of training a model $F_\mathcal{A}$ with high capacity. If the watermark is not embedded, we expect $F_\mathcal{A}$'s accuracy on the primary task to be too low to make it usable (**W1**).

Different adversaries $\mathcal{A}_i$ will have different datasets $D_{\mathcal{A}_i}$. Consequently, the trigger sets $T_{\mathcal{A}_i}$ selected by DAWN will also be different. Different surrogate models $F_{\mathcal{A}_i}$ will embed distinctive watermarks. Each watermark thus links to the API client identifier. DAWN meets the linkability requirement **W4**.

4.3 Watermark verification

We present the *verify* function used by $\mathcal{J}$ to prove a model F' is a surrogate of F. *Verify* tests if a given watermark $(T, \hat{B}(T))$ is embedded in a model F' suspected to be a surrogate of F. We first define $L(T, \hat{B}(T), F')$ that computes the ratio of different results between the backdoor function $\hat{B}(x)$ and the suspected surrogate model $\hat{F}'(x)$ for all inputs in the trigger set.

$$L(T, \hat{B}(T), F') = \frac{1}{|T|} \sum_{x \in T} (\hat{F}'(x) \neq \hat{B}(x)) \qquad (2)$$

The watermark verification succeeds, i.e., *verify* returns "true", if and only if $L(T, \hat{B}(T), F') < e$, where e is a tolerated error rate that must be defined. This means we must have at most $\lfloor e \times |T| \rfloor$ samples where $\hat{B}(x)$ and $\hat{F}'(x)$ differ in order to declare F' is a surrogate of F. The choice for the value of e is a trade-off between correctness and completeness for watermark verification (reliability **W2**). Assume we want to use a pre-generated watermark $(T, \hat{B}(T))$ to verify if an arbitrary model F' is a surrogate. For simplicity, we assume a uniform probability of matching the prediction of a watermarked input $P(\hat{B}(x) = \hat{F}'(x)) = 1/m$, where m is the number of classes of F'. The probability for trivial watermark verification success, given a trigger set of size $|T|$ and an error rate e, can be computed using the cumulative binomial distribution function as follows.

$$P(L < e) = \sum_{i=0}^{\lfloor e \times |T| \rfloor} \binom{|T|}{i} \times \left(\frac{m-1}{m}\right)^i \times \left(\frac{1}{m}\right)^{|T|-i} \qquad (3)$$

This probability is the average success rate of $\mathcal{A}$ wanting to frame $\mathcal{V}$ for model stealing using an arbitrary watermark. Figure 2 depicts

the decrease of this success rate as we increase the watermark size. We see that the verification function can accommodate a large error rate ($e > 0.5$) while preventing trivial success in verification using a small watermark ($|T| \approx 50$). The error rate e must be defined proportionally to the number of classes m. Large error rates can be used for models with a large number of classes. For instance, we can set $e = 0.8$ for a model with $m = 256$ classes, limiting the adversary success rate to less than 2^{-64} for a watermark of size 70.
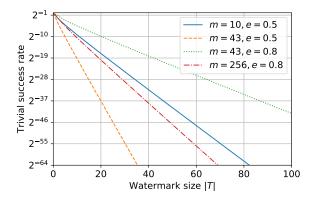

Figure 2: Resilience to trivial watermark verification vs. size of the watermark. Different tolerated error rates e and models with different number of classes m. The success rate for trivial watermark verification decreases exponentially with the watermark size.

The success rate in trivial verification is the complement of the confidence for reliable watermark verification, and for reliable demonstration of ownership by transition $1 - P(L < e)$. The choice of e defines the minimum watermark size given a targeted confidence. Recall that this size must also be small to ensure utility of the model to protect **X1**. The tolerated error must necessarily be lower than the probability of random class match: $e < (1 - m)/m$. Also, e must be larger than ϵ where $Acc(F_\mathcal{A}) = 1 - \epsilon$ is the accuracy of the watermarked surrogate model $F_\mathcal{A}$ on the trigger set.

The success of watermark verification is not sufficient to declare ownership of a surrogate model F'. $\mathcal{A}$ can increase its success in trivial watermark verification from random using several means. For instance, knowing F and F', $\mathcal{A}$ can find inputs x for which $F(x) \neq F'(x)$ and use pairs $(x, F'(x))$ as a watermark that would successfully pass watermark verification. Thus demonstrating ownership requires a careful process to ensure that the probability for matching an incorrect prediction class remains random, ensuring that the probability for trivial watermark verification follows Eq. 3.

4.4 Demonstrating ownership

We present the process for a model owner $\mathcal{V}$ to demonstrate ownership of a surrogate model watermarked by DAWN. It only requires the suspected surrogate model $F_\mathcal{A}$ to expose a prediction API. This process uses a judge $\mathcal{J}$ who is trusted to (a) ensure confidentiality of all data submitted as input to the process and (b) correctly execute and report the results of the specified *verify*. It also uses a time-stamped public bulletin board, e.g., a blockchain, in which

information can be published to provide proof of anteriority. $\mathcal{J}$ can be implemented using an trusted execution environment (TEE) [12].

4.4.1 Watermark registration. $\mathcal{V}$ publishes cryptographic commitments of the following elements in the public bulletin board:

- the model $F_\mathcal{V}$.
- for each API client i, one registered watermark $(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i}))$.

The commitment can be instantiated using a cryptographic hash function $H(\ )$, e.g., SHA-3. Each watermark should be linked to the corresponding model, e.g., by associating $H(F_\mathcal{V})$ with each registered watermark.

Several updated versions of the registered watermark can be published for each API client, as they make more queries to the prediction API and their watermarks grow. The verification of any one of these watermarks is sufficient to demonstrate ownership of the model. We define the following rules for reliable demonstration of ownership **W2** that prevents ownership piracy **W3**:

- $\left(H(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i})), H(F_\mathcal{V})\right)$ is valid only if published later than $H(F_\mathcal{V})$.
- $\mathcal{A}$ can refute $F_\mathcal{A}$ is a surrogate model only if $H(F_\mathcal{A})$ has been published.
- $\left(H(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i})), H(F_\mathcal{V})\right)$ can only demonstrate that $F_\mathcal{A}$ is a surrogate of $F_\mathcal{V}$ if $H(F_\mathcal{A})$ is published later than $H(F_\mathcal{V})$ (or not published at all).
- in case of contention, the model having its commitment first published is deemed to be the original.

4.4.2 Verification process. When $\mathcal{V}$ suspects a model $F_\mathcal{A}$ is a surrogate of $F_\mathcal{V}$ trained by an API client i, it provides a pointer to the prediction API of $F_\mathcal{A}$ to $\mathcal{J}$. It also provides the following secret information using a confidential communication channel: the API client i watermark $(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i}))$ and $F_\mathcal{V}$. $\mathcal{J}$ does the following to check if $F_\mathcal{A}$ is a surrogate of $F_\mathcal{V}$. If any step fails, the ownership of $F_\mathcal{A}$ is not considered to have been demonstrated. If all succeed, $\mathcal{J}$ gives the verdict that $F_\mathcal{A}$ is a surrogate of $F_\mathcal{V}$.

(1) compute $H(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i}))$ and use it as a pointer to retrieve the registered watermark $\left(H(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i})), H(F'_\mathcal{V})\right)$ from the public bulletin.
(2) compute $H(F_\mathcal{V})$ and verify $H(F_\mathcal{V}) = H(F'_\mathcal{V})$, where $H(F'_\mathcal{V})$ is extracted from the registered watermark.
(3) retrieve $H(F_\mathcal{V})$ from the public bulletin and verify it was published before $\left(H(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i})), H(F'_\mathcal{V})\right)$.
(4) query $T_{\mathcal{A}_i}$ to $F_\mathcal{A}$'s prediction API and verify that $L(T_{\mathcal{A}_i}, \hat{B}_\mathcal{V}(T_{\mathcal{A}_i}), F_\mathcal{A}) < e$.
(5) input $T_{\mathcal{A}_i}$ to $F_\mathcal{V}$ and verify $\hat{B}_\mathcal{V}(x) \neq \hat{F}_\mathcal{V}(x), \forall x \in T_{\mathcal{A}_i}$.

If $F_\mathcal{A}$'s owner ($\mathcal{A}$) wants to contest the verdict, it must provide the original model $F'_\mathcal{A}$ to $\mathcal{J}$ using a confidential communication channel. $\mathcal{J}$ assesses that the provided model and the API model are the same $F_\mathcal{A} = F'_\mathcal{A}$ by verifying $F_\mathcal{A}(x) = F'_\mathcal{A}(x), \forall x \in T_{\mathcal{A}_i}$. Then, $\mathcal{J}$ computes $H(F'_\mathcal{A})$ and retrieves it from the public bulletin. If $H(F'_\mathcal{A})$ was published before $H(F_\mathcal{V})$, $\mathcal{J}$ concludes that $F'_\mathcal{A} = F_\mathcal{A}$ is an original model.

5 EXPERIMENTAL SETUP

5.1 Datasets and Models

5.1.1 Datasets. We evaluate DAWN using four image recognition datasets that were used in prior work to evaluate DNN extraction attacks and are presented in App. A. MNIST [26] and GTSRB [39] are respectively a handwritten-digit and traffic-sign dataset used to showcase the extraction of low capacity DNN models [21, 34]. CIFAR10 [24] and Caltech256 [16] contain images depicting miscellaneous objects that were used to showcase the extraction of high capacity DNN models [9, 33].

We also selected a random subset of 100,000 samples from ImageNet dataset [11], which contains images of natural and man-made objects. We use it to evaluate the embedding of different types of watermarks and to perform a model extraction attack that requires such samples [33].

5.1.2 Models. We select two kinds of DNN models to evaluate the embedding of a watermark: low-capacity models having less than 10M parameters, and high-capacity models having over 20M parameters. These models are presented in Table 2 and their architecture is detailed in App. A.

Table 2: DNN models used to evaluate DAWN. Number of training epochs and base test accuracy.

Model	Input size	m	Param.	Epochs	Acc_{test}
MNIST-3L	28x28x1	10	62,346	10	98.6
MNIST-5L	28x28x1	10	683,522	10	99.1
GTSRB-5L	32x32x3	43	669,123	50	91.7
CIFAR10-9L	32x32x3	10	~ 6 M	100	84.6
GTSRB-RN34	224x224x3	43	~ 21 M	250	98.1
CIFAR10-RN34	224x224x3	10	~ 21 M	250	94.7
Caltech-RN34	224x224x3	256	~ 21 M	250	74.4

In order to accurately reconstruct model extraction attacks, we use the same model architectures and training process as in [21] for low-capacity models and as in [33] for high-capacity models. Similarly to prior work [33], we use ResNet34 [19] architecture pre-trained on ImageNet as a basis for high-capacity models. We fine-tuned Caltech-RN34, GTSRB-RN34 and CIFAR10-RN34 models using Caltech256, GTSRB and CIFAR10 datasets respectively [1]. We also trained DenseNet121 [20] models to perform additional experiments due to the absence of dropout layers in ResNet34 models. All models were trained using Adam optimizer with learning rate of 0.001 that was decreased over time to 0.0005 (after 100 epochs for ResNet34 models and half-way for the other), except for Caltech-RN34. For Caltech-RN34, we used SGD optimizer with an initial learning rate of 0.1 that was decreased by a factor of 10 every 60 epochs over 250 epochs. We used a batch size of 16 for fine-tuning ResNet34 and DenseNet121 based models.

---

[1]We chose to reproduce only the Caltech-RN34 experiment from [33] because of its best performance. We used CIFAR10 and GTSRB to conduct supplementary experiments with high capacity models as they allow us to juxtapose results of experiments with low and high capacity models on the same datasets.

5.2 Watermarking Procedure

Inputs from $\mathcal{A}$'s dataset $D_{\mathcal{A}}$ are submitted to the DAWN-enhanced prediction API of $F_{\mathcal{V}}$ which returns correct $F_{\mathcal{V}}(x)$ or incorrect predictions $B_{\mathcal{V}}(x)$ according to the result of the watermarking function $W_{\mathcal{V}}(x)$. For simplicity, we implement $M_{\mathcal{V}}(x)$ as an identity function in all our experiments. The evaluation and comparison of different mapping methods is outside the scope of this paper. We simulate $\mathcal{A}$ who uses the whole set $D_{\mathcal{A}}$, which includes $|T_{\mathcal{A}}|$ samples with incorrect labels, to train its surrogate model $F_{\mathcal{A}}$. $\mathcal{A}$ trains $F_{\mathcal{A}}$ without being aware of the watermarked samples in $D_{\mathcal{A}}$.

5.3 Evaluation Metrics

We use two metrics to evaluate the success of $\mathcal{A}$'s goal and $\mathcal{V}$'s goal respectively. $\mathcal{A}$'s goal is to train a surrogate model $F_{\mathcal{A}}$ that has maximum accuracy on $F_{\mathcal{V}}$'s primary classification task. We evaluate this by computing the *test accuracy* of the surrogate model $Acc_{test}(F_{\mathcal{A}})$ on the test set $Test$ of each dataset (cf. Tab. 7).

$$Acc_{test}(F_{\mathcal{A}}) = \frac{1}{|Test|} \sum_{x_i \in Test} \begin{cases} 1, & \text{if } \hat{F}_{\mathcal{A}}(x_i) = O_f(x_i) \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$\mathcal{V}$'s goal is to maximize the embedding of the watermark in any surrogate model built from responses from $F_{\mathcal{V}}$ such that its surrogacy can be reliably demonstrated. We evaluate this by computing the *watermark accuracy* of the surrogate model $Acc_{wm}(F_{\mathcal{A}})$ on the trigger set $T_{\mathcal{A}}$ of watermarked inputs.

$$Acc_{wm}(F_{\mathcal{A}}) = \frac{1}{|T_{\mathcal{A}}|} \sum_{x_i \in T_{\mathcal{A}}} \begin{cases} 1, & \text{if } \hat{F}_{\mathcal{A}}(x_i) = \hat{B}_{\mathcal{V}}(x_i) \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

DAWN aims to maximize $Acc_{wm}(F_{\mathcal{A}})$ regardless of $Acc_{test}(F_{\mathcal{A}})$. $\mathcal{A}$ aims to maximize $Acc_{test}(F_{\mathcal{A}})$ while minimizing $Acc_{wm}(F_{\mathcal{A}})$. In our experiments, we calculate both metrics every 5 epochs in order to evaluate their progress during the training process.

6 ROBUSTNESS OF WATERMARKING

We assess $\mathcal{A}$'s ability to prevent the embedding of a watermark in a surrogate model, i.e., to violate the unremovability requirement **W1**. Prior work evaluated unremovability *after* a watermarked model is trained showing that backoor-based watermarks are resilient to model pruning and adversarial fine tuning [1, 31, 48]. DAWN also embeds backdoor-based watermarks resilient to removal using post-training manipulations. Thus, we focus on adversarial manipulations *during* training by evaluating several solutions that could prevent watermark embedding. We then evaluate the ability for $\mathcal{A}$ to identify watermarked inputs using the trained surrogate model, i.e., to violate the indistinguishability requirement **X2**.

We take an ideal model extraction attack scenario where $\hat{F}_{\mathcal{V}} = O_f$ is a perfect oracle. $\mathcal{A}$ has access to a large dataset $D_{\mathcal{A}}$ of natural samples from the same distribution as $\mathcal{V}$ training data: we use the whole training set from each dataset (Tab. 7) for $D_{\mathcal{A}}$. We use a large watermark of fixed size $|T_{\mathcal{A}}| = 250$ in all following experiments. Embedding a large watermark is challenging since the model must learn many isolated errors (mislabeled inputs). We take $|T_{\mathcal{A}}| = 250$ as an upper bound to the watermark size and a worst case scenario for DAWN watermark embedding.

6.1 Unremovability of watermark during training

We evaluate the impact of two parameters on embedding a watermark during DNN training. The first parameter is the capacity of $F_{\mathcal{A}}$. $\mathcal{A}$ can limit this capacity such that the model could only learn the primary classification task and cannot learn the watermark. The second parameter is the use of regularization. Regularization accommodates classification errors on the training data, which is considered as noise. The watermark consists of incorrectly labeled inputs which may be discarded using regularization.

We evaluate the impact of model capacity and regularization on watermark accuracy Acc_{wm} and test accuracy Acc_{test} of $F_{\mathcal{A}}$. We trained several surrogate models having low and high capacity. $T_{\mathcal{A}}$ was randomly selected from the respective training sets. We used plain training and two regularization methods, namely weight decay [25] with decaying factor λ and dropout (DO=X) [38] with probability X={0.3, 0.5}. We selected λ values optimal for $\mathcal{A}$: such that they maximize the difference $Acc_{test} - Acc_{wm}$.

Table 3a and 3b present the results of this experiment for DNN models with low and high capacity respectively. We report Acc_{wm} and Acc_{test} results at three training stages providing (1) best watermark accuracy (best for $\mathcal{V}$), (2) best test accuracy (best for $\mathcal{A}$) and (3) when training is completed. Overall, we observe that Acc_{test} and Acc_{wm} are high for most settings. Using plain training, Acc_{wm} is mostly higher than Acc_{test} and often close to 100%. The ownership of all these surrogate models can be reliably demonstrated using a low tolerated error rate, e.g., $e = 0.3$.

Figure 3 depicts the evolution of Acc_{wm}, Acc_{test} and training loss during the training of some selected surrogate models. Using plain training, we see that watermark and test accuracy are closely tied and Acc_{wm} is usually slightly higher than Acc_{test}.

It is also worth noting that training a watermarked model is slower than training a plain model. Training an accurate watermarked MNIST-5L model requires 100 epochs while training the same model without watermark requires 10 epochs (cf. Tab. 2). Nevertheless this difference in training time cannot be exploited by $\mathcal{A}$ to infer if the predictions it gets are watermarked or not. $\mathcal{A}$ does not have an expected baseline training time prior to extract a victim model. We can see that training time for a surrogate model depends on the victim model, which is unknown to $\mathcal{A}$ (100 epochs for MNIST-5L / 20 epochs for CIFAR-9L in Fig. 3).

Model capacity. High-capacity models can provide higher watermark and test accuracy than low-capacity models as highlighted by comparing results for GTSRB and CIFAR10 in both tables. While Acc_{wm} is low for some low-capacity models, e.g., MNIST-3L, MNIST-5L (DO), their test accuracy is similarly low and close to random $Acc_{wm} \sim Acc_{test} \sim 10\%$. This shows that reducing the model capacity can prevent the embedding of the watermark. However, decreasing Acc_{wm} to a level where it cannot be used to reliably prove ownership makes $F_{\mathcal{A}}$ unusable. Acc_{wm} and Acc_{test} are closely tied when manipulating the model capacity and thus this is not a useful strategy to circumvent DAWN.

Regularization. Regularization is useful for decreasing the watermark accuracy in a few cases. Weight decay is useful for low-capacity GTSRB-5L and CIFAR10-9L models. Dropout is useful
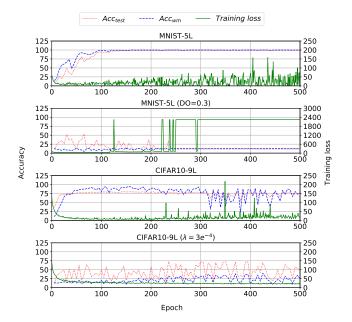
Figure 3: Evolution of training loss, test (Acc_{test}) and watermark accuracy (Acc_{wm}) over 500 training epochs. MNIST-5L and CIFAR10-9L are baseline models using plain training. Acc_{test} and Acc_{wm} are tied and vary simultaneously. The trained surrogate model embeds the watermark. MNIST-5L (DO=0.3) and CIFAR10-9L ($\lambda = 3e^{-4}$) respectively use dropout and weight decay. Acc_{test} can be significantly higher than Acc_{wm} but its evolution is very unstable and it has large variations despite the training loss remaining low. Obtaining a usable non-watermarked surrogate model is challenging.

for low-capacity MNIST-5L and CIFAR10-9L models, and for high-capacity Caltech-DN121 model. Dropout completely prevents the embedding of the watermark into MNIST-5L model as depicted by $Acc_{wm} \sim 10\%$. However, Acc_{test} is also significantly reduced, by 50% at best, making $F_{\mathcal{A}}$ potentially unusable. Figure 3 (MNIST-5L (DO=0.3)) further highlights that the maximal test accuracy of 51% is difficult to obtain since Acc_{test} is very unstable along the training process, changing abruptly from 10% to 50% while the training loss remains constantly low. In all remaining cases, Acc_{wm} is reduced down to 20-35%, while preserving high test accuracy similar to models trained with non-watermarked datasets. While Acc_{wm} is low, the watermark can still successfully demonstrate ownership by increasing the tolerated error rate to, e.g., $e = 0.8 > 1 - Acc_{wm}$. Considering the large watermark size of 250, this demonstration would still be reliable despite the high tolerated error rate as evaluated in Sect. 4.3.

It is worth noting that no regularization method is effective at removing the watermark from high capacity GTSRB-RN34 and CIFAR10-RN34 models. The likely reason is that ResNet34 architecture has significant overcapacity for the primary task of classifying these datasets. Regularization cannot limit this capacity to an extent

Table 3: Impact of regularization – dropout (DO) and weight decay (λ) – on test ($test$) and watermark accuracy (wm) of surrogate models $F_{\mathcal{A}}$. We report results at the training epoch (ep.) reaching best Acc_{wm} (optimal for $\mathcal{V}$), best Acc_{test} (optimal for $\mathcal{A}$) and when training is over (Final). Green results highlight low Acc_{wm} and Acc_{test}: $F_{\mathcal{A}}$ is unusable. Red results highlight low Acc_{wm} while Acc_{test} remains significantly high: $\mathcal{V}$ may fail to prove ownership of $F_{\mathcal{A}}$.

(a) Low capacity models. 500 training epochs.

Model	Best Acc_{wm}			Best Acc_{test}			Final	
	wm	test	ep.	wm	test	ep.	wm	test
MNIST-3L	14%	11%	–	14%	11%	–	14%	11%
MNIST-3L (DO=0.3)	12%	11%	–	12%	11%	–	12%	11%
MNIST-3L (DO=0.5)	13%	11%	–	13%	11%	–	13%	11%
MNIST-3L ($\lambda = 5e^{-6}$)	99%	89%	210	98%	96%	290	97%	96%
MNIST-5L	99%	96%	120	98%	97%	170	98%	97%
MNIST-5L (DO=0.3)	13%	16%	50	11%	51%	30	12%	11%
MNIST-5L (DO=0.5)	13%	17%	15	9%	53%	50	11%	13%
MNIST-5L ($\lambda = 5e^{-6}$)	99%	88%	215	99%	94%	365	98%	93%
GTSRB-5L	97%	88%	160	95%	89%	190	97%	88%
GTSRB-5L (DO=0.3)	99%	88%	135	98%	90%	220	98%	88%
GTSRB-5L (DO=0.5)	98%	89%	105	98%	90%	200	98%	89%
GTSRB-5L ($\lambda = 5e^{-6}$)	28%	55%	410	17%	71%	105	25%	79%
CIFAR10-9L	93%	78%	110	92%	79%	105	73%	76%
CIFAR10-9L (DO=0.3)	40%	75%	125	35%	75%	90	25%	70%
CIFAR10-9L (DO=0.5)	45%	71%	240	25%	77%	90	25%	75%
CIFAR10-9L ($\lambda = 3e^{-4}$)	32%	72%	235	32%	72%	235	25%	47%

(b) High capacity models. 250 training epochs.

Model	Best Acc_{wm}			Best Acc_{test}			Final	
	wm	test	ep.	wm	test	ep.	wm	test
GTSRB-RN34	83%	97%	245	70%	98%	105	84%	97%
GTSRB-DN121 (DO=0.3)	98%	89%	240	98%	89%	240	95%	86%
GTSRB-DN121 (DO=0.5)	99%	92%	235	98%	93%	245	98%	93%
GTSRB-RN34 ($\lambda = e^{-5}$)	87%	92%	200	87%	92%	200	73%	77%
CIFAR10-RN34	99%	89%	110	99%	90%	240	98%	89%
CIFAR10-DN121 (DO=0.3)	99%	88%	160	98%	88%	210	97%	86%
CIFAR10-DN121 (DO=0.5)	99%	85%	130	97%	88%	220	98%	87%
CIFAR10-RN34 ($\lambda = e^{-5}$)	100%	80%	10	100%	89%	160	97%	81%
Caltech-RN34	97%	69%	110	93%	73%	160	94%	73%
Caltech-DN121 (DO=0.3)	48%	44%	110	36%	59%	155	32%	57%
Caltech-DN121 (DO=0.5)	35%	30%	115	22%	49%	185	21%	49%
Caltech-RN34 ($\lambda = 3e^{-4}$)	89%	67%	100	69%	68%	60	76%	68%

where the watermark would not be embedded. This means $\mathcal{A}$ needs sufficient knowledge of $F_{\mathcal{V}}$ to select an appropriate model architecture for $F_{\mathcal{A}}$. It must have sufficient capacity to learn the primary classification task of the victim model while preventing watermark embedding. In model extraction attacks, $\mathcal{A}$ has black-box access to $F_{\mathcal{V}}$, which forces to use $F_{\mathcal{A}}$ with sufficient capacity to maximize the attack success [33]. In this setting, regularization is not useful to circumvent DAWN.

Finally, while regularization can be useful, $\mathcal{A}$ needs relevant test data and ground truth to optimize the regularization parameters (e.g., decaying factor λ). Also, we observed in Fig. 3 that test accuracy is very unstable while the training loss remains constantly low when using regularization. $\mathcal{A}$ needs additional labeled test data to apply early stopping [5] of training at the optimal epoch providing

Table 4: Resistance of $M_\mathcal{V}$ to perturbations δ of various size for the MNIST dataset.

	Entire $D_\mathcal{A}$			$T_\mathcal{A}$ only		
	same $\hat{F}_\mathcal{V}$		diff $\hat{F}_\mathcal{V}$	same $\hat{F}_\mathcal{V}$		diff $\hat{F}_\mathcal{V}$
δ	same $M_\mathcal{V}$	diff $M_\mathcal{V}$		same $M_\mathcal{V}$	diff $M_\mathcal{V}$	
0.2	99.30%	0.44%	0.26%	73.88%	13.55%	12.57%
0.1	99.63%	0.24%	0.13%	85.12%	7.52%	7.36%
0.09	99.64%	0.22%	0.14%	85.70%	8.01%	6.29%
0.075	99.71%	0.19%	0.10%	88.84%	4.55%	6.61%
0.05	99.81%	0.12%	0.07%	92.98%	3.97%	3.05%

maximal test accuracy. In all extraction attacks [9, 21, 33, 34, 42] the availability of relevant data is the main limitation. All this data is typically used for training the surrogate model and none is used for test purposes, which prevents optimization of regularization parameters and early stopping.

A third parameter that may impact watermark embedding is the distribution of watermarked inputs. In contrast to prior DNN watermarking solutions[1, 31, 48], DAWN uses watermarked inputs from the same distribution as the training data. The watermark distribution cannot be controlled by $\mathcal{A}$.

6.2 Detection of Watermarked Inputs

$\mathcal{A}$ can try to identify watermarked inputs and remove them from $D_\mathcal{A}$ prior to training in order to prevent watermark embedding. Because DAWN relies on a hash to decide if an input is watermarked, $\mathcal{A}$ can query multiple perturbed versions of inputs in $D_\mathcal{A}$ and discard those that return different predictions. The mapping function $M_\mathcal{V}$ presented in Sect. 4.1.3 is meant to prevent this evasion. We evaluate the effectiveness of $M_\mathcal{V}$ by querying 10 perturbed versions of each of the 10,000 samples in $D_\mathcal{A}$, which includes $|T_\mathcal{A}| = 121$. For each query, we check whether they get consistent mapping $M_\mathcal{V}$ and classification $\hat{F}_\mathcal{V}$. Table 4 reports the results of this experiment for various perturbation size δ for the MNIST dataset. We distinguish cases where ❶ $\hat{F}_\mathcal{V}(x) = \hat{F}_\mathcal{V}(x + \delta)$ (same $\hat{F}_\mathcal{V}$) or $\hat{F}_\mathcal{V}(x) \neq \hat{F}_\mathcal{V}(x + \delta)$ (diff $\hat{F}_\mathcal{V}$); ❷ $M_\mathcal{V}(x) = M_\mathcal{V}(x + \delta)$ (same $M_\mathcal{V}$) or $M_\mathcal{V}(x) \neq M_\mathcal{V}(x + \delta)$ (diff $M_\mathcal{V}$). Same $\hat{F}_\mathcal{V}$ and same $M_\mathcal{V}$ means $\mathcal{A}$ keeps a watermarked sample in $D_\mathcal{A}$ ($M_\mathcal{V}$ succeeds). Same $\hat{F}_\mathcal{V}$ and different $M_\mathcal{V}$ means $\mathcal{A}$ discards a watermarked sample from $D_\mathcal{A}$ ($M_\mathcal{V}$ fails). Different $\hat{F}_\mathcal{V}$ means $\mathcal{A}$ wrongfully discards a sample from $D_\mathcal{A}$ regardless of $M_\mathcal{V}$ (δ is too large and changes $F_\mathcal{V}$'s prediction). We see $M_\mathcal{V}$ succeeds to provide a consistent mapping in over 85% cases for $\delta \leq 0.1$, meaning 85% of the $T_\mathcal{A}$ is preserved in $D_\mathcal{A}$. As perturbations δ increase in size, $M_\mathcal{V}$ returns an increasing rate of inconsistent mapping, but this rate is similar to the one of changed predictions from $F_\mathcal{V}$. Thus, we conclude $M_\mathcal{V}$ is resilient to perturbations and DAWN can effectively watermark $F_\mathcal{A}$.

Alternatively, several techniques can identify if a DNN model has a backdoor [6, 18, 45]. Most techniques like *Neural Cleanse* [45] and *TABOR* [18] can only detect backdoors for which the trigger is a static pattern added to original inputs (e.g., yellow square added to an image). In contrast, our trigger set is composed of unmodified samples having only incorrect labels. Consequently, technique like *Neural Cleanse* and *TABOR* are ineffective at detecting DAWN's watermark. In App. B, we evaluate one technique [6] that claims to

be effective at identifying a trigger set $T_\mathcal{A}$ in the entire training set $D_\mathcal{A}$. We show that DAWN's watermarks are undetectable by this technique, which is not more effective than randomly discarding some portion of $D_\mathcal{A}$.

7 PROTECTING AGAINST MODEL EXTRACTION ATTACKS

We evaluate DAWN's effectiveness at watermarking surrogate DNN models constructed using two model extraction attacks presented in App. C. The PRADA attack [21] achieves state-of-the-art performance in extracting low-capacity DNN models primarily using synthetic data and we launch it against MNIST-5L, GTSRB-5L and CIFAR10-9L. The KnockOff attack [33] extracts high-capacity DNN models using only natural data and we launch it against GTSRB-RN34, CIFAR10-RN34 and Caltech-RN34. The test accuracy of each $F_\mathcal{A}$ extracted with these respective attacks is reported in Tab. 6.

We demonstrate how to setup DAWN to protect a given victim model $F_\mathcal{V}$. We evaluate the successful embedding of watermarks in several surrogate models $F_\mathcal{A}$ as well as their utility considering a circumvention strategy.

7.1 Effectiveness of DAWN

Watermarking decision: DAWN degrades $F_\mathcal{V}$ utility by a factor equal to $r_w \times Acc(F_\mathcal{V})$ due to incorrect predictions for watermarked inputs. The value of r_w is specific to $F_\mathcal{V}$. Given a desired level of confidence for reliable ownership demonstration equal to $1 - P(L < e)$ (cf. Eq. 3), a tolerated error rate e and the number of classes m for $F_\mathcal{V}$, we can compute the minimum size for the watermark $|T_\mathcal{A}|$ using Eq. 3. Given that $\mathcal{V}$ can estimate the minimum number of queries N required by $\mathcal{A}$ to train a usable surrogate model for $F_\mathcal{V}$, we can compute $r_w = N/|T_\mathcal{A}|$. This ratio ensures that if $\mathcal{A}$ can successfully train a usable surrogate model $F_\mathcal{A}$, then $F_\mathcal{A}$ will embed a watermark large enough to reliably demonstrate its ownership .

The probability for successful trivial watermark verification $P(L < e)$ is valid for testing a single watermark. This probability increases by a factor equal to the number of tested watermarks. DAWN creates and registers client-specific watermarks. $\mathcal{V}$ must estimate the number of API clients to calculate the actual probability for trivial demonstration of ownership considering that all registered watermarks should be tested. When verifying a watermark, the judge $\mathcal{J}$ counts the number of registered watermarks for $F_\mathcal{V}$ in the public bulletin. $\mathcal{J}$ computes the real probability for successful trivial watermark verification accordingly and decides if a demonstration of ownership is reliable or not according to this final confidence.

Utility for legitimate clients: Suppose we want a confidence for reliable demonstration of ownership equal to $1 - 2^{-64}$. $F_\mathcal{V}$ has a prediction API with 1M API clients (1M watermarks are registered for $F_\mathcal{V}$). We need $P(L < e) < 10^{-6} \times 2^{-64} = 5.4 \times 10^{-26}$ to be able to test all registered watermarks while achieving our targeted confidence. We choose a tolerated error rate $e = 0.5$. Table 5 reports the computed watermark ratio r_w required to protect six models against model extraction. We see r_w must always be lower than 0.5% to reach $1 - 2^{-64}$ confidence for any victim model. $F_\mathcal{V}$'s accuracy is thus degraded in a negligible manner that does not impact its utility. DAWN meets the reliability **W2** and utility **X1** requirements.

Table 5: Ratio of watermarked inputs r_w required to protect six victim models F_V from extraction attack (PRADA for 3 first models / KnockOff for 3 last). Prediction API with 1M clients and targeted confidence for reliable demonstration of ownership = $1 - 2^{-64}$. Number of attack queries (N) obtained from [21, 33] and used to compute the watermark size $|T_A|$. F_V test accuracy decreases in a negligible manner ($r_w < 0.5\%$) that does not impact its utility.

Model	classes	queries (N)	$|T_A|$	r_w(%)	New $Acc(F_V)$
MNIST-5L	10	25,600	109 (0.1MB)	0.426	98.7%
GTSRB-5L	43	25,520	47 (0.4MB)	0.184	91.5%
CIFAR10-9L	10	160,000	109 (0.6MB)	0.068	84.5%
GTSRB-RN34	43	100,000	47 (1.7MB)	0.047	98.1%
CIFAR10-RN34	10	100,000	109 (3.9MB)	0.109	94.6%
Caltech-RN34	256	100,000	27 (1.0MB)	0.027	74.4%

Table 6: Efficacy of DAWN to defend against PRADA and KnockOff model extraction attacks. Baseline gives the test accuracy Acc_{test} of the victim F_V and surrogate model F_A trained without DAWN in place. F_A with DAWN provides Acc_{test} and watermark accuracy Acc_{wm} of F_A when DAWN protects F_V. All F_A have high $Acc_{wm} > 0.6$ allowing successful demonstration of ownership.

		Baseline Acc_{test}		F_A with DAWN	
Attack	Model	F_V	F_A	Acc_{test}	Acc_{wm}
	MNIST-5L	98.71%	95%	78.93%	100.00%
PRADA	GTSRB-5L	91.50%	61.00%	61.43%	98.23%
	CIFAR10-9L	84.53%	60.03%	60.95%	71.17%
	GTSRB-RN34	98.42%	97.43%	97.72%	100.00%
KnockOff	CIFAR10-RN34	94.66%	88.27%	88.41%	100.00%
	Caltech-RN34	74.62%	72.74%	71.98%	93.54%

Overhead: Storing 1M watermarks would require at most a few TBs (cf. Tab. 5). Watermark verification consists in obtaining predictions from a purported surrogate model. It is operated by J who gets predictions at no monetary cost. Thus, demonstration of ownership is only a matter of time and getting one prediction from our most complex model (Caltech-RN34) takes 9ms (on Tesla P100 GPU). Verifying one watermark for this model takes 0.25s (27 queries) and verifying 100,000 watermarks takes 7 hours using a single GPU. J can initially verify all watermarks with a lower confidence to reduce this time (by testing only a subset of each watermark). Only successful verification would later undergo a verification of the full watermark. Testing the same 100,000 watermarks with $1 - 2^{-16}$ targeted confidence (instead of $1 - 2^{-64}$) requires 1h15 (5 samples per watermark). This time can further be reduced by parallelizing predictions on several GPUs.

7.2 Effectiveness against real extraction attacks

We want to show that any surrogate F_A of a victim model F_V protected by DAWN will embed a watermark that allows for reliable demonstration of ownership. We evaluate the effectiveness of DAWN against two landmark model extraction attacks namely PRADA and KnockOff, presented in Sect. C.

Low-capacity models expose a prediction API that returns prediction classes $\hat{F}_V$ required for the PRADA attack. High-capacity models return the full probability vector F_V. Each victim model is protected by DAWN using the setting presented in Sect. 7.1. This setting enables V to demonstrate ownership of each surrogate model with confidence $1 - 2^{-64}$ using a tolerated error rate $e = 0.5$. For demonstration of ownership to be successful, the surrogate model F_A must pass the watermark verification test $L(T_A, \hat{B}_V(T_A), F_A) < e$. In our setting, it means that DAWN successfully defends against an extraction attack if the watermark accuracy for F_A is larger than 50%, i.e., $Acc_{wm}(F_A) > 1 - e$.

Table 6 presents the result of this experiment. We see all surrogate models have a watermark accuracy $Acc_{wm} \geq 66\%$, which means V is successful in demonstrating their ownership. DAWN successfully defends against the PRADA and KnockOff attacks for all tested models while incurring little decrease in F_V's utility (evaluated in Sect. 7.1). It is also worth noting that in all cases except for MNIST-5L, DAWN does not degrade the test accuracy of F_A.

We have shown DAWN effectively embeds a watermark in surrogate models F_A stolen using extraction attacks. In Appendix D, we evaluate the resilience of DAWN's watermark to removal using three attacks: (1) double-extraction of a *second order surrogate model* F'_A, (2) fine-tuning [23] and (3) pruning [4]. We show that pruning removes the watermark while sacrificing the test accuracy for the surrogate model. Double-extraction and fine-tuning can remove the watermark while preserving test accuracy given that A has unlimited access to natural data. However, DAWN defeats all three attacks when A has limited access to data, which is the case for most model extraction attack scenarios (cf. Sect. 2.2). A's access to data is limited in many scenarios, e.g., medical imaging classifiers, where these removal attacks would be ineffective.

7.3 Resilience to distributed extraction attack

Distributing a model extraction attack across several API clients means several adversaries A_i query a subset D_{A_i} from the whole set D_A used to train the surrogate model F_A. Recall that DAWN is a deterministic mechanism Sect. 4.1. The watermarking W_V and backdoor B_V functions are deterministic and specific to F_V. Their results only depend on the input queried to F_V. The responses to D_A, and its corresponding trigger set, remain the same regardless of which client(s) query the prediction API. Thus, D_A is labeled in the same manner and it includes the same trigger set T_A whether it is queried by one or by multiple API clients. Thus, the watermark in F_A trained using D_A will remain indistinguishable **X2** and unremovable **W1** even if multiple clients collude.

Note that in the case of colluding clients, each adversary A_i has a subset T_{A_i} of the whole trigger set T_A. When verifying ownership, the judge J will have several successful watermark verifications $L(T_{A_i}, B_V(T_{A_i}), F_A) < e$: one for each adversary A_i who colluded to build the surrogate model F_A. The verification of each subwatermark $(T_{A_i}, B_V(T_{A_i}))$ has the same expectation for success as the verification of the whole watermark $(T_A, B_V(T_A))$. J will conclude that each API client i whose watermark is successfully verified is a perpetrator of the distributed extraction attack used to build the surrogate model F_A. Linkability **W4** remains valid in case of collusion.

In a distributed attack, the watermark associated to each colluding client is smaller than in a centralized attack. To verify ownership with a same reliability, we must increase the watermark size and consequently r_w by a factor equal to the number of colluding clients. We assume the number of real colluding clients is limited, e.g., a few tens. Nevertheless, it is possible to mount a Sybil attack in which several API accounts are created by a single adversary. The API account registration process must require providing information that maximizes difficulty of creating trusted accounts, e.g., verified phone number or credit card, to mitigate this threat. Also, Sybils-detection techniques exist [43, 46] and it is possible to link Sybils accounts by examining querying patterns and IP addresses for instance [40]. For example, to protect Caltech-RN34, we could increase r_w to reliably verify the watermark of 35 colluders while maintaining the utility loss below 1%.

8 DISCUSSION

In Appendix F, we summarize how DAWN satisfies requirements introduced in Sec. 3.4. Here we discuss limitations of DAWN.

$\mathcal{A}$ can attempt to prevent ownership demonstration for $F_{\mathcal{A}}$ by ensuring that watermark verification (Eq. 2) fails. This entails reducing the watermark accuracy by training $F_{\mathcal{A}}$ using only a subset of the trigger set $T_{\mathcal{A}}$. Since watermarked inputs are indistinguishable (X2) and uniformly distributed in $D_{\mathcal{A}}$, $\mathcal{A}$ cannot selectively discard them. Nevertheless, $\mathcal{A}$ can discard $x\%$ of the whole $D_{\mathcal{A}}$, which statistically, will result in $x\%$ of watermarked input being discarded. This should reduce $Acc_{wm}(F_{\mathcal{A}})$ by $100 - x\%$. If x is high enough, the resulting $Acc_{wm}(F_{\mathcal{A}})$ can be brought down low enough for watermark verification to systematically fail.

While this strategy is effective, it deprives $\mathcal{A}$ from a large part of $D_{\mathcal{A}}$. This decreases $Acc_{test}(F_{\mathcal{A}})$ and consequently the utility of $F_{\mathcal{A}}$ (Appendix D). Alternatively, $\mathcal{A}$ must collect a set $D_{\mathcal{A}}$ $x\%$ larger and make $x\%$ more queries to $F_{\mathcal{V}}$ to compensate for later discarded training inputs. We already discussed in Sect 2.2 that *access to relevant data is the main limitation* for $\mathcal{A}$. The secondary goal of $\mathcal{A}$ is to limit the number of queries to $F_{\mathcal{V}}$ (cf. Sect. 3.1). This evasion strategy requires more adversarial capabilities (access to data) and it compromises one adversary goal (minimum number of queries). Consequently, even if effective, we do not consider it a realistic evasion strategy.

Another potential limitation of DAWN is circumvention of the mapping function $M_{\mathcal{V}}$ (Sec. 4.1.3). If mapping is too aggressive, $\mathcal{A}$ may probe the input space and try to identify subspaces that are grouped together. However, this is not guaranteed to work because the behavior of the model on synthetic samples is undefined [15]. $M_{\mathcal{V}}$ impacts only the watermarking decision $W_{\mathcal{V}}$ and not the returned label - $\mathcal{A}$ cannot interact directly with the mapping function. On the other hand, if the tolerated modification δ is too small, $\mathcal{A}$ might identify watermarked queries by submitting several samples with minor modifications and taking the majority vote of the label. We evaluated this attack in Sect. 6.2 showing that mappings from $M_{\mathcal{V}}$ are as consistent as predictions from $F_{\mathcal{V}}$. Semantic-preserving modifications to image queries (e.g., translation, rotation, change in color intensity, etc,) could be used to improve this attack. However, by using an embedding from $F_{\mathcal{V}}$ to implement $M_{\mathcal{V}}$, both functions should be as resilient to semantic-preserving modifications.

Finally, $\mathcal{A}$ can try to detect when $\mathcal{J}$ is making queries to demonstrate ownership by checking if they are part of $D_{\mathcal{A}}$ used to steal $F_{\mathcal{V}}$. This search can be implemented efficiently by building a hash table of $D_{\mathcal{A}}$. However, in Appendix E, we show that we can query noisy versions of samples in $T_{\mathcal{A}}$ while maintaining $Acc_{wm}(F_{\mathcal{A}})$ (c.f. Table 13). To identify these samples, $\mathcal{A}$ needs to compute their distance to every samples in $D_{\mathcal{A}}$. For a $D_{\mathcal{A}}$ containing 100,000 ImageNet samples, this computation is over 100 times more expensive than checking a hash table and it can take over 7 seconds per query, which is not feasible (c.f. Table 14).

9 RELATED WORK

Watermarking DNN models. The first watermarking technique for DNNs [44] explicitly embeds additional information into the weights of a DNN after it is trained. Verifying the watermark requires white-box access to the model in order to analyze the weights. A limitation of this approach is that the watermark can be easily removed by minimally retraining the watermarked model.

Alternative approaches [1, 10, 31, 48] that are more robust have been proposed, where the watermark can be verified in a black-box setting. These are based on backdooring and they allow for watermark extraction using only a prediction API, as discussed in Sect. 2.3. These approaches use both a carefully selected trigger set and a specific training process chosen by the model owner. The first proposal for such approach [31] consist in modifying the original model boundary using adversarial retraining [29] in order to make the model unique. The watermark is composed of synthetically generated adversarial samples [15] that are close to the decision boundary. The impact of selecting a particular distribution for a watermark has been evaluated in [48]. It shows that selecting a trigger set from the same distribution as the training data (albeit with minor synthetic modifications) or from a different distribution, does not affect the accuracy of the model for its primary classification task or on its training time, while the watermark gets perfectly embedded. Finally, more formal foundations and theoretical guarantees for backdoor-based DNN watermarking have been provided in [1]. This work empirically assesses that the removability of a DNN watermark is highly dependent on the training process of the watermarked model (training from scratch vs. re-training).

In contrast to prior DNN watermarking techniques for black-box verification [1, 10, 31, 48], DAWN considers a victim who (a) does not control the training of the DNN model and (b) cannot select a trigger set T from the whole input space $\mathbb{R}^n$. DAWN dynamically embeds a watermark in queries made to a model prediction API. Thus, DAWN defends against model extraction attacks and enables a model owner to identify surrogates of its model.

Defenses against model extraction. It was suggested that the distribution of queries made during an extraction attack is different from benign queries [21]. Hence, model extraction can be detected using density estimation methods, namely by assessing the ability for queries to fit a Gaussian distribution or not. However, this technique protects only against attacks using synthetic queries and is not effective against, e.g., the KnockOff attack. Other detection methods analyse subsequent queries close to the classes' decision boundaries [36] or queries exploring abnormally large region of

the input space [22]. Both methods are effective but detect only extraction attacks against decision trees. They are ineffective against complex models like DNNs. Altering predictions returned to API clients can mitigate model extraction attacks. Predictions can be restricted to classes [42] or adversarially modified to degrade the performance of the surrogate model [27]. However, some extraction attacks [21] circumvent such defenses because they remain effective using just prediction classes.

Prior defenses to model extraction are designed to protect only simple models [22, 36] or to prevent only specific extraction attacks [27]. It is arguable if a generic defense would ever be effective at detecting/preventing model extraction. Consequently, with DAWN we take a different approach where we assume a surrogate model can be extracted. Then we propose a generic defense to identify surrogate DNN models that have been extracted from any victim model using any extraction attack.

REFERENCES

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium*. 1615–1631.
[2] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*. 1467–1474.
[3] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. Springer.
[4] Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the State of Neural Network Pruning? *ArXiv* abs/2003.03033 (2020).
[5] Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*. 402–408.
[6] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2019. Detecting backdoor attacks on deep neural networks by activation clustering. In *AAAI Workshop on Artificial Intelligence Safety (SafeAI)*. 1–8.
[7] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).
[8] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918* (2019).
[9] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. 2018. Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[10] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. DeepSigns: An End-to-End Watermarking Framework for Ownership Protection of Deep Neural Networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 485–497.
[11] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. 2009. Imagenet: A large-scale hierarchical image database. In *In CVPR*.
[12] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. 2014. The Untapped Potential of Trusted Execution Environments on Mobile Devices. *IEEE Security & Privacy* 12, 4 (2014).
[13] Ronald Aylmer Fisher, Frank Yates, et al. 1949. Statistical tables for biological, agricultural and medical research. *Statistical tables for biological, agricultural and medical research.* Ed. 3. (1949).
[14] Forbes. 2019. Roundup Of Machine Learning Forecasts And Market Estimates For 2019. https://www.forbes.com/sites/louiscolumbus/2019/03/27/roundup-of-machine-learning-forecasts-and-market-estimates-2019. (2019). Online; accessed 9 May 2019.
[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[16] Gregory Scott Griffin, Alex Holub, and Pietro Perona. 2007. Caltech-256 Object Category Dataset.
[17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).
[18] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. 2019. TABOR: A Highly Accurate Approach to Inspecting and Restoring Trojan Backdoors in AI Systems. *arXiv preprint arXiv:1908.01763* (2019).

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*. 4700–4708.
[21] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. 2019. PRADA: Protecting against DNN Model Stealing Attacks. In *IEEE European Symposium on Security & Privacy*. IEEE, 1–16.
[22] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. 2018. Model Extraction Warning in MLaaS Paradigm. In *34th Annual Computer Security Applications Conference*.
[23] Simon Kornblith, Jonathon Shlens, and Quoc V Le. 2018. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974* (2018).
[24] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.
[25] Anders Krogh and John A Hertz. 1992. A simple weight decay can improve generalization. In *Advances in neural information processing systems*. 950–957.
[26] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist. *AT&T Labs* (2010).
[27] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. 2018. Defending Against Model Stealing Attacks Using Deceptive Perturbations. *arXiv preprint arXiv:1806.00054* (2018).
[28] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *Network and Distributed Systems Security Symposium*. 1–15.
[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
[30] Dongyu Meng and Hao Chen. 2017. MagNet: A Two-Pronged Defense Against Adversarial Examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 135–147. https://doi.org/10.1145/3133956.3134057
[31] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2017. Adversarial frontier stitching for remote neural network watermarking. *arXiv preprint arXiv:1711.01894* (2017).
[32] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *ACM Workshop on Artificial Intelligence and Security*. ACM, 27–38.
[33] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff Nets: Stealing Functionality of Black-Box Models. In *CVPR*. 4954–4963.
[34] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *ACM Symposium on Information, Computer and Communications Security*. ACM, 506–519.
[35] Fabien AP Petitcolas, Ross J Anderson, and Markus G Kuhn. 1999. Information hiding-a survey. *Proc. IEEE* 87, 7 (1999), 1062–1078.
[36] E. Quiring, D. Arp, and K. Rieck. 2018. Forgotten Siblings: Unifying Attacks on Machine Learning and Digital Watermarking. In *IEEE European Symposium on Security & Privacy*. 488–502.
[37] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.
[38] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[39] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2011. The German traffic sign recognition benchmark: a multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*.
[40] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. 2015. EVILCOHORT: Detecting Communities of Malicious Accounts on Online Services. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 563–578. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/stringhini
[41] TechWorld. 2018. How tech giants are investing in artificial intelligence. https://www.techworld.com/picture-gallery/data/tech-giants-investing-in-artificial-intelligence-3629737. (2018). Online; accessed 9 May 2019.
[42] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium*. 601–618.
[43] Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. 2009. Sybil-resilient Online Content Voting. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*. USENIX Association, Berkeley, CA, USA, 15–28.
[44] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *ACM International Conference*

on Multimedia Retrieval. ACM, 269–277.

[45] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *IEEE Symposium on Security & Privacy.*

[46] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y. Zhao. 2013. You Are How You Click: Clickstream Analysis for Sybil Detection. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13).* USENIX, Washington, D.C., 241–256. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/wang

[47] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. https://arxiv.org/abs/1611.03530

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

A DATASETS AND MODELS

Table 7 presents the characteristics of the datasets we used in our experiments. These are divided into a training and a testing set. Images were resized to fit the corresponding model architectures used in prior work. Table 8 presents model architectures used for conducting experiments with low capacity models - the perfect-knowledge attacker in Sect. 6.1 and reproduction of the PRADA [21] attack in Sect. 7.2.

Table 7: Image datasets used to evaluate DAWN. Different sample sizes are input to different models.

| | | | Number of samples | |
Dataset	Sample Size	Classes	Train	Test
MNIST	28x28	10	60,000	10,000
GTSRB	32x32 / 224x224	43	39,209	12,630
CIFAR10	32x32 / 224x224	10	50,000	10,000
Caltech	224x224	256	23,703	6,904
ImageNet	224x224	1000	100,000	-

B DETECTING WATERMARKED INPUTS

We assess if watermarked inputs can be identified such that $\mathcal{A}$ could remove them from the $D_{\mathcal{A}}$ before training the surrogate model.

This defense consists in first training a DNN model with the whole training dataset. Then, training data is predicted using the trained model and we record the activations of the last hidden layer of the DNN model. These activations are projected to three dimensions using Independent Component Analysis (ICA) and clustered into two clusters using k-means. These clusters are expected to group benign training data and poisoned data (watermarked inputs) respectively. The intuition for this approach is that incorrectly labeled inputs (watermark) trigger different activations than correctly labeled inputs in the trained DNN model. The size and silhouette score [37] of the two clusters are analyzed to conclude (1) if there is backdoor in the model and (2) which training inputs compose the backdoor. According to authors, a low silhouette score (0.1/0.15) and a high difference in relative cluster size is expected if the model embeds a watermark. The smallest cluster should contain the watermarked inputs.

To evaluate this defense against DAWN, we trained two sets of DNN models, plain models using a correctly labeled training set only, and watermarked models each embedding a watermark of size

Table 8: Model architectures of low capacity models.

Layer	MNIST-3L	MNIST-5L	GTSRB-5L	CIFAR10-9L
1	conv2-32	conv2-32	conv2-64	conv2-32
	maxpool2d	maxpool2d	maxpool2d	batchnorm2d
	ReLU	ReLU	ReLU	ReLU
2	conv2-64	conv2-64	conv2-128	conv2-64
	maxpool2d	maxpool2d	maxpool2d	ReLU
	ReLU	ReLU	ReLU	maxpool2d
3	dropout	conv2-128	dropout	conv2-128
	FC-10	maxpool2d	FC-200	batchnorm2d
	Softmax	ReLU	ReLU	ReLU
4		dropout	dropout	conv2-128
		FC-200	FC-100	ReLU
		ReLU	ReLU	maxpool2d
5		dropout	dropout	dropout
		FC-10	FC-43	conv2-256
		Softmax	Softmax	batchnorm2d
				ReLU
6				conv2-256
				ReLU
				maxpool2d
7				dropout
				FC-1024
				ReLU
8				FC-512
				ReLU
9				dropout
				FC-10
				Softmax

Table 9: Results of watermark detection [6] on several watermarked (*wm*) and plain models (No *wm*). Relative size represents the average ratio of training inputs ($D_{\mathcal{A}}$) contained in the small cluster (supposed to contain watermarked inputs only). *wm* split counts watermarked inputs in the small/large clusters ($|T_{\mathcal{A}}| = 250$). Silhouette score [37] is averaged over all classes. Small clusters are much larger than the size of the watermark. Most watermarked inputs are contained in large clusters. Silhouette score for both plain and watermarked models is above the recommended detection threshold (0.15). DAWN's watermark cannot be detected.

| | Relative size | | *wm* | Silhouette score | |
Model	*wm*	No *wm*	split	*wm*	No *wm*
MNIST-5L	0.222	0.449	55/195	0.47 ± 0.33	0.23 ± 0.06
GTSRB-5L	0.059	0.098	59/191	0.64 ± 0.18	0.76 ± 0.20
CIFAR10-9L	0.094	0.105	3/247	0.79 ± 0.18	0.78 ± 0.19
GTSRB-RN34	0.409	0.419	92/158	0.26 ± 0.09	0.24 ± 0.02
CIFAR10-RN34	0.378	0.338	79/171	0.24 ± 0.01	0.27 ± 0.04
Caltech-RN34	0.425	0.422	124/126	0.24 ± 0.02	0.24 ± 0.02
Overall	0.264	0.305	67/183	0.44	0.42

$|T_{\mathcal{A}}| = 250$. We applied the watermark detection process discussed above on these models and report results in Tab. 9. Clustering is not
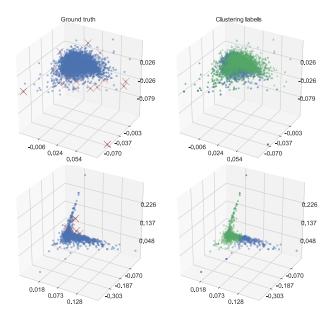
Figure 4: Last activation (three independent components) for inputs predicted "1" by MNIST-5L watermarked model (top) and "4" by GTSRB-5L watermarked model (bottom). Left: Ground truth watermarked inputs (red cross) and correctly labeled inputs (blue dots). Right: Clustering results of watermark detection (two clusters: blue dots / green triangles). Watermarked inputs are mixed with correctly labeled inputs and clusters cannot isolate the watermark.

able to isolate watermarked inputs into a single cluster; the main part of watermarked inputs belongs to large clusters. The recommendation [6] to discard small clusters from training would deprive $D_{\mathcal{A}}$ from a large number of correctly labeled samples while a large part of the watermark would be preserved. Using this approach, the defense wrongly discards 26.4% of clean data from $D_{\mathcal{A}}$ on average while detecting only 67 out of 250 watermarked samples (26.8% of $T_{\mathcal{A}}$). The silhouette score is not useful for detecting the watermark either since watermarked and plain models have close scores that are all above the recommended detection threshold (0.1/0.15) [6]. Our plain models are detected as embedding a watermark using this defense. We conclude that this defense is ineffective at detecting watermarks generated by DAWN.

We assume the reason for this ineffectiveness is due to the nature of our watermark, selected from the same distribution as the training set. In contrast to prior DNN watermarking solutions [1, 31, 48], our watermarked inputs do not come from a single manifold distant from the training data manifold. Consequently the model does not learn a "single" activation that generalizes to the whole watermark but rather learns individual exceptions for each watermarked input. The activations of watermarked inputs are thus different from each other and they are scattered among the activations of the remaining training data (correctly labeled). This can be observed in Fig. 4 - left, where we see that watermarked inputs are scattered among

correctly labeled inputs. This explains why generated clusters cannot isolate watermarked inputs from correctly labeled inputs (Fig. 4 - right).

C MODEL EXTRACTION ATTACKS CONSIDERED

We selected two model extraction attacks to evaluate DAWN.

PRADA is an iterative model extraction attack that consists of several *duplication rounds*. Each round is composed of three steps: ❶ querying $F_{\mathcal{V}}$, ❷ training $F_{\mathcal{A}}$ with obtained predictions $F_{\mathcal{V}}(x)$ and ❸ crafting new synthetic queries using the updated $F_{\mathcal{A}}$. $\mathcal{A}$ first queries $F_{\mathcal{V}}$ with natural queries called *seed samples*. Then, it crafts synthetic queries for which predictions are used to refine $F_{\mathcal{A}}$'s decision boundary. This means $D_{\mathcal{A}}$ is built iteratively as $F_{\mathcal{A}}$ is retrained and refined. PRADA uses prediction classes $\hat{F}_{\mathcal{V}}(x)$ to train $F_{\mathcal{A}}$. We launch the PRADA attack against low-capacity models MNIST-5L, GTSRB-5L and CIFAR10-9L. Following the setup of Juuti et al. [21], we use the same model architecture for both $F_{\mathcal{V}}$ and $F_{\mathcal{A}}$, and $F_{\mathcal{A}}$ is trained from scratch. We use 10 natural seed samples per class to attack MNIST-5L and GTSRB-5L, we perform 8 and 6 duplication rounds respectively, using respectively *TRND* and *COLOR* synthetic sample generation strategy. PRADA is not evaluated against CIFAR10-9L in prior work [21]. We use 1000 natural seed samples per class, 4 duplication rounds using *TRND* to train $F_{\mathcal{A}}$ having a satisfactory accuracy.

KnockOff is a model extraction attack to steal image classification DNN models. It queries natural image inputs coming from a distribution that is independent of the distribution of $\mathcal{V}$'s training set. All queries ($D_{\mathcal{A}}$) and their predictions are used to retrain (fine-tune) a high-capacity pre-trained DNN model. Knockoff uses full probability vector $F_{\mathcal{V}}(x)$ to retrain the surrogate model. We launch the KnockOff attack against high-capacity models GTSRB-RN34, CIFAR10-RN34 and Caltech-RN34. Following the setup of Orekondy et al. [33], we use ResNet34 architecture with ImageNet pre-trained weights as basis for $F_{\mathcal{A}}$. Then, $F_{\mathcal{A}}$ is fine-tuned using 100,000 queries ($D_{\mathcal{A}}$) sampled from the ImageNet dataset. For fine tuning, we use stochastic gradient descent (SGD) with momentum of 0.5 for 200 epochs. We chose an initial learning rate of 0.1 which is decayed every 60 epochs by a factor of 10.

As it is observed in Tab. 6, surrogate models have a test accuracy that is significantly lower (15-30 percentage points -pp) than their victim counterparts in all cases except for MNIST-5L and GTSRB-RN34. Model extraction attacks induce a decrease in accuracy [21, 33, 34] and our results are comparable to those obtained in prior work [21, 33].

D WATERMARK REMOVAL

Double extraction and fine-tuning. A watermark may be removed by performing an extraction attack against $F_{\mathcal{A}}$ to obtain a second order surrogate model $F'_{\mathcal{A}}$. $\mathcal{A}$ has full control over $F_{\mathcal{A}}$: its prediction API is not protected by DAWN and does not intentionally return incorrect prediction. If $\mathcal{A}$ uses a disjoint set of queries to extract a surrogate $F'_{\mathcal{A}}$ from $F_{\mathcal{A}}$, then $F'_{\mathcal{A}}$ may not embed the watermark, preventing the demonstration of its ownership by $\mathcal{V}$. Instead of starting the second extraction from scratch, it can use $F_{\mathcal{A}}$ as the

starting point for $F_{\mathcal{A}'}$ and fine-tune it. We call this *stealing+fine-tuning*.

We observed in Tab. 6 that surrogate models have a lower accuracy than victim models because model extraction incurs a necessary decrease in surrogate model accuracy. We evaluate the extent of the decrease in Acc_{test} and Acc_{wm} if $\mathcal{A}$ launches two successive extraction attacks instead of one or steals+fine-tunes the model to obtain $F'_{\mathcal{A}}$: the first against $F_{\mathcal{V}}$ and the second against $F_{\mathcal{A}}$. We evaluate these evasion techniques using the PRADA and KnockOff attacks.

The two successive extraction attacks and stealing+fine-tuning are performed in the same conditions as described in Sect. C. The only difference is that $\mathcal{A}$ uses half the seed samples for each PRADA attack (5 per class for MNIST-5L and GTSRB-5L, 500 per class for CIFAR10-9L) and runs an additional duplication round to query the same number of inputs. The number of seed samples is a limited adversarial capability in PRADA [21], so we grant $\mathcal{A}$ with the same capability for single and double extraction attack. For each KnockOff attack, $\mathcal{A}$ uses a different set of 100,000 inputs from ImageNet. For the second extraction attack (against $F_{\mathcal{A}}$), we query the same number of inputs as for the first one. While this number can be increased, we empirically observed that the test accuracy of $F'_{\mathcal{A}}$ reaches its maximum and stagnates before the PRADA and KnockOff attacks finish, i.e., more queries do not improve Acc_{test} of $F'_{\mathcal{A}}$.

Table 10: Double extraction attack: test accuracy Acc_{test} for the victim model $F_{\mathcal{V}}$, first order surrogate model $F_{\mathcal{A}}$ and second order surrogate model $F'_{\mathcal{A}}$. Watermark accuracy Acc_{wm} for $F'_{\mathcal{A}}$ and Increase in Acc_{test} degradation (Degrad.) between $F_{\mathcal{A}}$ and $F'_{\mathcal{A}}$ compared to between $F_{\mathcal{V}}$ and $F_{\mathcal{A}}$. Double extraction can remove DAWN's watermark from $F'_{\mathcal{A}}$ but it incurs a significant additional degradation in test accuracy (20-80%). Test accuracy of $F'_{\mathcal{A}}$ for the PRADA attack is too low for it to be useful.

| | $F_{\mathcal{V}}$ | $F_{\mathcal{A}}$ | $F'_{\mathcal{A}}$ (2nd extraction) | | |
Model	Acc_{test}	Acc_{test}	Acc_{test}	Acc_{wm}	Degrad.
MNIST-5L	98.7%	71.10%	49.06% (-49pp)	11.11%	+80%
GTSRB-5L	91.5%	50.66%	29.51% (-62pp)	3.36%	+52%
CIFAR10-9L	84.5%	45.7%	37.80% (-44pp)	3.04%	+20%
GTSRB-RN34	98.42%	97.72%	97.30% (-1pp)	17.64%	+71%
CIFAR10-RN34	94.66%	88.41%	84.85% (-10pp)	11.88%	+57%
Caltech-RN34	74.62%	71.98%	70.95% (-4pp)	9.67%	+39%

As it can be observed in Tab. 10 and 11, double extraction attack and stealing+fine-tuning effectively remove the watermark from the second order surrogate model $F'_{\mathcal{A}}$. The watermark accuracy is low enough (3-22%) to fail demonstration of ownership for $F'_{\mathcal{A}}$, which empirically confirms that prior DNN watermarking techniques are not resilient to this class of model extraction attacks [48].

While removing the watermark, the extraction of the second order surrogate model using these attacks also increases the degradation in test accuracy by 20% to 80% for $F'_{\mathcal{A}}$ compared to $F_{\mathcal{A}}$. Considering a powerful $\mathcal{A}$ having unlimited access to natural data (e.g., KnockOff adversary model) the extraction of the first order

Table 11: Stealing+fine-tuning attack: test accuracy Acc_{test} for the victim model $F_{\mathcal{V}}$, surrogate model $F_{\mathcal{A}}$ and fine-tuned surrogate model $F'_{\mathcal{A}}$. Watermark accuracy Acc_{wm} for $F'_{\mathcal{A}}$ and increase in Acc_{test} degradation (Degrad.) between $F_{\mathcal{A}}$ and $F'_{\mathcal{A}}$ compared to between $F_{\mathcal{V}}$ and $F_{\mathcal{A}}$. Fine-tuning can remove DAWN's watermark from $F'_{\mathcal{A}}$ but it incurs additional degradation in test accuracy (20-60%). Test accuracy of $F'_{\mathcal{A}}$ for the PRADA attack is too low for it to be useful.

| | $F_{\mathcal{V}}$ | $F_{\mathcal{A}}$ | $F'_{\mathcal{A}}$ (fine-tuning) | | |
Model	Acc_{test}	Acc_{test}	Acc_{test}	Acc_{wm}	Degrad.
MNIST-5L	98.7%	71.10%	63.67% (-35pp)	22.22%	+27%
GTSRB-5L	91.5%	50.66%	42.09% (-49pp)	9.24%	+21%
CIFAR10-9L	84.5%	45.7%	38.41% (-46pp)	4.06%	+19%
GTSRB-RN34	98.42%	97.72%	97.22% (-1pp)	7.81%	+60%
CIFAR10-RN34	94.66%	88.41%	84.93% (-10pp)	16.83%	+56%
Caltech-RN34	74.62%	71.98%	70.81% (-4pp)	22.58%	+44%

surrogate model incurs little accuracy degradation and so does the extraction of the second order surrogate model. The final model $F'_{\mathcal{A}}$ stolen using KnockOff attack has its watermark removed and preserves its utility (from -1pp to -10pp compared to $F_{\mathcal{V}}$). DAWN cannot protect model extraction attacks where $\mathcal{A}$ has unlimited access to natural data. However, $\mathcal{A}$'s access to data is limited in many scenarios, e.g., access to medical imaging that are privacy sensitive, and highly specialized models may not return meaningful predictions to random images. In such a scenario, the KnockOff attack may not be effective and the PRADA attack is more effective. We see that for the PRADA attack the test accuracy of $F'_{\mathcal{A}}$ decreases sharply during each attack that removes the watermark (3 top rows in Tab. 10 and 11) . In most cases, the final test accuracy of $F'_{\mathcal{A}}$ is less than half of $F_{\mathcal{V}}$ (from -35pp to -62pp) and we consider that $Acc(F'_{\mathcal{A}}) \ll Acc(F_{\mathcal{V}})$ makes $F'_{\mathcal{A}}$ too inaccurate to be useful. DAWN can effectively protect against extraction attack using a limited amount of data by destroying the utility of a model $F'_{\mathcal{A}}$ deprived from watermark.

Pruning. Alternatively, $\mathcal{A}$ may attempt to prune the model by setting random weights of the model to zero. In our experiments we prune weights uniformly randomly. We show that for large values of δ pruning is effective at removing the watermark but it sacrifices models utility and renders it useless (c.f. Table 12). Furthermore, we observe that Acc_{test} and Acc_{wm} do not fall proportionally i.e. there is no guarantee that sacrificing X% Acc_{test} results in the same drop in Acc_{wm}. Also, our experiments show that determining appropriate δ without knowing $T_{\mathcal{A}}$ is challenging as there is no consistent drop in accuracy for a particular value of δ across all models.

E NOISY WATERMARK VERIFICATION

$\mathcal{A}$ having deployed $F_{\mathcal{A}}$ and being aware of DAWN can try to prevent watermark verification performed by $\mathcal{J}$. $\mathcal{A}$ can check if queries to $F_{\mathcal{A}}$ belong to $D_{\mathcal{A}}$ used to steal $F_{\mathcal{V}}$, and return different predictions for them. We evaluated that searching $D_{\mathcal{A}}$ for exact matches can be done efficiently using a hash table (28-44ms additional overhead per query for our datasets). Hashing the query is the most time consuming part of this search.

Table 12: Acc_{test} and Acc_{wm} of pruned stolen models $F_{\mathcal{A}}$. δ denotes the percentage of weights set to zero.

Model		δ (%)					
		0	25	40	50	75	90
MNIST-5L	Acc_{test}	78.93%	78.57%	78.31%	77.76%	71.24%	43.84%
	Acc_{wm}	100.00%	100.00%	100.00%	100.00%	61.50%	7.69%
GTSRB-5L	Acc_{test}	61.43%	54.24%	43.80%	30.57%	3.81%	3.80%
	Acc_{wm}	98.23%	90.26%	50.44%	20.35%	3.53%	3.53%
CIFAR10-9L	Acc_{test}	60.95%	59.83%	48.45%	44.96%	10.00%	10.00%
	Acc_{wm}	71.17%	70.20%	54.91%	16.58%	16.26%	11.43%
GTSRB-RN34	Acc_{test}	97.72%	97.72%	97.71%	97.41%	45.73%	0.47%
	Acc_{wm}	72.54%	62.74%	50.98%	33.34%	11.62%	1.96%
CIFAR10-RN34	Acc_{test}	88.41%	87.85%	87.12%	86.48%	36.69%	10.00%
	Acc_{wm}	100.00%	99.00%	91.08%	81.18%	7.92%	4.95%
Caltech-RN34	Acc_{test}	71.98%	71.06%	68.48%	63.55%	4.92%	0.33%
	Acc_{wm}	93.54%	93.54%	70.96%	61.29%	6.45%	0.00%

Table 13: Acc_{wm} when querying $T_{\mathcal{A}}$ with added ϵ perturbation during verification. $Acc_{wm} > 50\%$ ensures successful demonstration of ownership.

Model / ϵ	Acc_{wm}						
	0.0	0.1	0.25	0.4	0.5	0.75	0.9
MNIST-5L	100%	100%	100%	100%	92.30%	53.84%	30.76%
GTSRB-5L	98.23%	96.46%	88.49%	66.37%	37.16%	10.61%	4.42%
CIFAR10-9L	71.17%	69.56%	51.36%	31.56%	19.80%	7.89%	7.56%
GTSRB-RN34	72.54%	66.67%	54.90%	29.41%	21.56%	11.76%	9.80%
CIFAR10-RN34	100%	99.00%	92.07%	67.32%	39.60%	19.08%	8.91%
Caltech-RN34	93.54%	90.32%	51.61%	19.35%	6.45%	0.00%	0.00%

Table 14: Time to find the closest sample in the dataset (ms), average over 10,000 tests. Exact lookup (Eq.) using a hash table is done in negligible time ($\approx 30ms$). Calculating distances has a substantial time overhead.

Dataset	samples	size	Eq.	L_0	L_2	L_{inf}
MNIST	60,000	28x28x1	28	230	770	570
GTSRB	39,209	32x32x3	29	172	420	420
CIFAR10	50,000	32x32x3	30	243	625	615
Caltech	23,703	224x224x3	30	351	2020	2040
ImageNet	100,000	224x224x3	44	1421	7362	7370

To prevent this evasion by $\mathcal{A}$, $\mathcal{J}$ can slightly perturb samples in $T_{\mathcal{A}}$ before submitting them for verification to $F_{\mathcal{A}}$. In Tab. 13, we show the resilience of watermark verification to various amounts of noise (ϵ) added to each image in $T_{\mathcal{A}}$. We show that we can maintain $Acc_{wm} > 50\%$ for all models up to $\epsilon = 0.25$ or even more for some models. $\mathcal{A}$ can no longer perform a simple lookup in a hash table to identify queries from $\mathcal{J}$ if $T_{\mathcal{A}}$ is perturbed. $\mathcal{A}$ must compute the distance to every sample in $D_{\mathcal{A}}$ and find the nearest neighbor to the query. We demonstrate that such search incurs a substantial computational overhead (c.f. Tab. 14). Searching in 100,000 ImageNet samples can take over 7s per query, which is too long to be acceptable for deployment. Also, a small distance to an element in $D_{\mathcal{A}}$ does not mean that queried image was in fact part of $D_{\mathcal{A}}$. $\mathcal{A}$ has to set up a threshold for rejecting the queries that will affect the utility of the model, on top of the described computational overhead.

F MEETING SYSTEM REQUIREMENTS

Unremovability W1. We extensively evaluated (Sect 6.1) that manipulation of the training process of $F_{\mathcal{A}}$ either does not prevent the embedding of the watermark or if it does, it significantly degrades $F_{\mathcal{A}}$'s test accuracy. Proper use of regularization can effectively mitigate the watermark embedding but it requires $\mathcal{A}$ to be granted more capabilities e.g. increased access to relevant data. We also showed (Sect. D) that two successive extraction attacks can remove a watermark from $F_{\mathcal{A}}$. However, it also decreases the test accuracy to an extent that makes $F_{\mathcal{A}}$ unusable. Finally, prior work [1, 31, 48] has shown that manipulations after training such as pruning and adversarial fine tuning are ineffective against backdoor-based DNN watermarks. We can conclude that DAWN watermarking meets unremovability.

Indistinguishability X2. We defined model-specific watermarking and backdoor functions (Sect. 4.1) that *always* return the same same output (correct or incorrect) for the same input. We also introduced a solution for mapping inputs with minor differences to similar predictions (Sect 4.1.3). Finally, we empirically assessed (Sect. B) that watermarks generated by DAWN are not detected by a recent defense against watermarking/backdooring. Hence, DAWN's watermarks are indistinguishable by API clients.

Reliability W2 and utility X1. The watermark registration and verification protocol that we introduce (Sect. 4.4) ensures that the success of $\mathcal{A}$ in demonstrating ownership of an arbitrary model is negligible. We showed how to set up DAWN in order to reliably demonstrate ownership of several surrogate models stolen using two state-of-the-art model extraction attacks with high confidence equal to $1 - 2^{-64}$ (Sect. 7). DAWN effectively watermarked every surrogate model $F_{\mathcal{A}}$ while causing a negligible decrease of $F_{\mathcal{V}}$'s utility (0.03-0.5%). DAWN allows for reliable ownership demonstration while preserving the utility.

Non-ownership piracy W3 is guaranteed by our watermark registration and verification protocol (Sect. 4.4). In case of contention, the first registered model is deemed the original.

Linkablity W4. DAWN selects watermarked inputs from API client queries and registers one watermark per API client. Different clients make different queries and they will consequently have different watermarks. Given that we meet the reliability requirement **W2**, a single watermark $(T_{\mathcal{A}_i}, B_{\mathcal{V}}(T_{\mathcal{A}_i}))$ will succeed in proving $F_{\mathcal{A}_i}$ is a surrogate of $F_{\mathcal{V}}$. Watermarks are API client-specific which makes a surrogate model linkable to an API client.

Collusion resistance X3. DAWN relies on deterministic functions for watermarking ($W_{\mathcal{V}}$) and backdooring ($B_{\mathcal{V}}$) that are specific to $F_{\mathcal{V}}$ but independent of the client sending a query. Thus, the watermark remains indistinguishable despite collusion **X2**. DAWN is resilient to a Sybil attack (bounded to a certain number of Sybils) and assure successful verification by $\mathcal{J}$ (Sect. 7.3).