DAWN: Dynamic Adversarial Watermarking of Neural Networks

Sebastian Szyller
Aalto University
contact@sebszyller.com

Buse Gul Atli
Aalto University
buse.atlitekgul@aalto.fi

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

N. Asokan
University of Waterloo & Aalto University
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 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 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 theft. Unlike prior watermarking schemes, DAWN does not impose changes to the training process but 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%).

CCS CONCEPTS

• **Security and privacy** → **Systems security**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Deep Neural Network; Watermarking; Model stealing; Model extraction; IP protection

ACM Reference Format:
Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. 2021. DAWN: Dynamic Adversarial Watermarking of Neural Networks. In *Proceedings of the 29th ACM Int'l Conference on Multimedia (MM '21), Oct. 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3474085.3475591

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 [35]. 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, 24, 41] 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* [36] 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 [15, 25], and are difficult to prevent [15]. 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, 41]. 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 (our code is available on GitHub[1]):

- 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 manipulation during training (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 Model Extraction Attacks

In model extraction [5, 15, 25, 27, 28, 36], 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 [5, 15, 25, 28, 36] 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, hyperparameters, or the training process of $F_{\mathcal{V}}$. However, given the purpose of the API (e.g., image recognition) and expected complexity of the task, $\mathcal{A}$ may guess the architecture of the model [15, 28].

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 [5]. If $\mathcal{A}$ does not exactly know the distribution or the domain, it may use widely available natural data [25, 27] to mount the attack. Alternatively, it may use only *synthetic samples* [36] or a mix of a small number of natural samples augmented by synthetic samples [15, 28].

Querying strategy. All model stealing attacks [5, 15, 25, 27, 28, 36] consist of alternating phases of $\mathcal{A}$ querying $F_{\mathcal{V}}$, followed by training the surrogate model $F_{\mathcal{A}}$ using the obtained predictions.

2.2 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 is currently based on backdooring attacks [1, 6, 24, 41]. 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* [25]: 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 [25] or synthetic [15, 28, 36]. $\mathcal{A}$ 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.

Assumptions. For a given input $x \in D_{\mathcal{A}}$, $\mathcal{A}$ has no a priori expectation regarding the prediction $F_{\mathcal{V}}(x)$. $\mathcal{A}$ treats $y = F_{\mathcal{V}}(x)$ 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 $\mathcal{V}$'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 [29].

3.2 DAWN Goals and Overview

We design a solution to identify and prove the ownership of DNN models stolen through a prediction API (see Fig. 1 in [34]). DAWN (Dynamic Adversarial Watermarking of Neural Networks), is an additional component added in front of a model prediction API (Sect. 3.2). 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 $B(x_i) \neq$

[1] github.com/ssg-research/dawn-dynamic-adversarial-watermarking-of-neural-networks

$F_{\mathcal{V}}(x_i)$. $\mathcal{A}$ 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 [3] 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 $F_{\mathcal{V}}$. *Verify* checks that for sufficient number of inputs $x_i \in T$, we have $\hat{F}_{\mathcal{A}}(x_i) = \hat{B}(x_i) \neq \hat{F}_{\mathcal{V}}(x_i)$. 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.

3.3 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 $F_{\mathcal{V}}$ is free of the watermark, then $Acc(F_{\mathcal{A}}) \ll Acc(F_{\mathcal{V}})$.
- **W2 Reliability**: If *verify* outputs "true" for a watermark $(T, \hat{B}_{\mathcal{V}}(T))$ on a model F', then F' is a surrogate of $F_{\mathcal{V}}$, 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 $\mathcal{V}$, such that it can cast $\mathcal{V}$'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 $F_{\mathcal{A}}$.

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: $Acc(\text{DAWN} + F_{\mathcal{V}}) \sim Acc(F_{\mathcal{V}})$.
- **X2 Indistinguishability**: $\mathcal{A}$ cannot distinguish incorrect predictions $B(x)$ from correct victim model predictions $F_{\mathcal{V}}(x)$.
- **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.

4 DYNAMIC ADVERSARIAL WATERMARKS

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.

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:

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

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.

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 [9] 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.

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. 3.1). Nevertheless, our watermarking function $W_{\mathcal{V}}$ uses a hash of the input x. A subtle modification δ to x produces a different $W_{\mathcal{V}}(x + \delta) \neq W_{\mathcal{V}}(x)$. 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 $M_{\mathcal{V}} : \mathbb{R}^n \rightarrow \mathbb{R}^p$ 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.

$M_{\mathcal{V}}$ 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 [23]. $M_{\mathcal{V}}$ could also be a masking and binning function [4] removing large modifications of a single pixel value (with masking) and small modifications of a large number of pixels (with binning). We evaluated one particular implementation of $M_{\mathcal{V}}$: 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. 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 [40]. This phenomenon is called overfitting and it can be prevented using regularization [2]. But it is not effective for DNNs with a large capacity [40]. This is the rationale for the existence of DNN backdoors [21] 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.

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

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.

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

This probability is the average success rate of $\mathcal{A}$ wanting to frame $\mathcal{V}$ for model stealing using an arbitrary watermark. Fig 2 in [34] 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.

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) [8].

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**:

- $\big(H(T_{\mathcal{A}_i}, \hat{B}_{\mathcal{V}}(T_{\mathcal{A}_i})), H(F_{\mathcal{V}})\big)$ 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.
- $\big(H(T_{\mathcal{A}_i}, \hat{B}_{\mathcal{V}}(T_{\mathcal{A}_i})), H(F_{\mathcal{V}})\big)$ 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 $\big(H(T_{\mathcal{A}_i}, \hat{B}_{\mathcal{V}}(T_{\mathcal{A}_i})), H(F'_{\mathcal{V}})\big)$ 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 $\big(H(T_{\mathcal{A}_i}, \hat{B}_{\mathcal{V}}(T_{\mathcal{A}_i})), H(F'_{\mathcal{V}})\big)$.
(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

Datasets. We evaluate DAWN using four image recognition datasets that were used in prior work to evaluate DNN extraction attacks. MNIST [19] (60,000 train and 10,000 test samples, 10 classes) and GTSRB [32] (39,209 train and 12,630 test samples, 43 classes) are respectively a handwritten-digit and traffic-sign dataset used to showcase the extraction of low capacity DNN models [15, 28]. CIFAR10 [17] (50,000 train and 10,000 test samples, 10 classes) and Caltech256 [11] (23,703 train and 6,904 test samples, 256 classes) contain images depicting miscellaneous objects that were used to showcase the extraction of high capacity DNN models [5, 25].

We also selected a random subset of 100,000 samples from ImageNet dataset [7] (1000 classes), 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 [25].

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 Tab. 1.

Table 1: Characteristics of models used in our evaluation.

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

Similarly to prior work [25], we use ResNet34 [12] 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. We also trained DenseNet121 [13] models to perform additional experiments due to the absence of dropout layers in ResNet34 models.

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 the experiments in Sect. 6.2 (evaluating the effectiveness of the mapping function $M_{\mathcal{V}}$), we use the embedding from $F_{\mathcal{V}}$ as $M_{\mathcal{V}}$. Experiments in Sect. 6.1 and Sect. 7 do not depend on the choice of $M_{\mathcal{V}}$. Therefore, for the sake of simplicity, we use the identity function as $M_{\mathcal{V}}$ in these experiments. 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 $D_{\mathcal{A}}$.

Evaluation metrics. The success of $\mathcal{A}$'s goal and $\mathcal{V}$'s goal are assessed using the *accuracy* metric. $\mathcal{A}$'s goal is to train a surrogate model $F_{\mathcal{A}}$ that has maximum accuracy on $F_{\mathcal{V}}$'s primary classification task i.e. high *test accuracy* $Acc_{test}(F_{\mathcal{A}})$ on the test set $Test$.

$\mathcal{V}$’s goal is to maximize the embedding of the watermark in any $F_{\mathcal{A}}$ built from responses from $F_{\mathcal{V}}$ such that its ownership can be demonstrated i.e. to have high *watermark accuracy* of the surrogate model $Acc_{wm}(F_{\mathcal{A}})$ on the trigger set $T_{\mathcal{A}}$ of watermarked inputs.

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**. We focus on adversarial manipulations *during* training by evaluating several solutions that could prevent watermark embedding. We also empirically evaluate that DAWN is resilient to watermark removal *after* training. This evaluation is included in the extended version of this work [34]. 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 $\mathcal{V}$’s training data: we use the whole training set from each dataset (Sect. 5.1) for $D_{\mathcal{A}}$. We use a large watermark of fixed size $|T_{\mathcal{A}}| = 250$.

Table 2: Impact of regularization – dropout (DO) and weight decay (λ) – on test (*test*) and watermark (*wm*) accuracy of $F_{\mathcal{A}}$. We report results at the training epoch (ep.) reaching best Acc_{wm} (optimal for $\mathcal{V}$), best Acc_{test} (optimal for $\mathcal{A}$), and at completion (Final). Purple (underline) highlights low Acc_{wm} and Acc_{test}: $F_{\mathcal{A}}$ is unusable. Red (dashed underline) highlights low Acc_{wm} ($< 50\%$) while Acc_{test} remains significantly high (decrease $< 10pp$): success for $\mathcal{A}$.

	Best Acc_{wm}			Best Acc_{test}			Final	
Model	*wm*	*test*	ep.	*wm*	*test*	ep.	*wm*	*test*
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 ($\lambda = 5e^{-6}$)	$\underline{28\%}$	$\underline{55\%}$	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 ($\lambda = 3e^{-4}$)	$\underline{32\%}$	$\underline{72\%}$	235	32%	72%	235	$\underline{25\%}$	$\underline{47\%}$
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%

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 can potentially be discarded using regularization.

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 (weight decay [18] with decaying factor λ and dropout (DO=X) [31] with probability X=$\{0.3, 0.5\}$). Tab. 2 shows Acc_{wm} and Acc_{test} at three training stages providing (1) best watermark accuracy (best for $\mathcal{V}$), (2) best test accuracy (best for $\mathcal{A}$), and (3) training is completion.

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. 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 for CIFAR10-9L models, and for high-capacity Caltech-DN121 model. 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. 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 [25].

6.2 Mapping Function

$\mathcal{A}$ can try to identify watermarked inputs by querying 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 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 $\hat{F}_{\mathcal{V}}$. Tab. 3 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

Table 3: $M_{\mathcal{V}}$'s resistance vs. perturbation size δ (MNIST).

	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%

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 $F_{\mathcal{A}}$.

We show that our approach is resilient to perturbations of various sizes in MNIST. In Tab. 3, we report results for T only ($|T| = 121$) and for the entire test set (size $10,000$). We distinguish 2 different cases: 1) $M_{\mathcal{V}}(x) = M_{\mathcal{V}}(x+\delta)$ and $F_{\mathcal{V}}(x) = F_{\mathcal{V}}(x+\delta)$ - neither prediction nor hash change; 2) $M_{\mathcal{V}}(x) \neq M_{\mathcal{V}}(x+\delta)$ and $F_{\mathcal{V}}(x) = F_{\mathcal{V}}(x+\delta)$ - hash changes but the prediction stays the same. Remainder corresponds to queries that fooled the model into making a wrong prediction which would result in a different $W_{\mathcal{V}}$ anyway.

7 PROTECTING AGAINST MODEL EXTRACTION ATTACKS

We evaluate DAWN's effectiveness at watermarking surrogate models constructed using two model extraction attacks: 1) the PRADA attack [15] 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; 2) the KnockOff attack [25] 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 attacks is reported in Tab. 5. We further discuss the effectiveness of DAWN in the extended version of this work [34].

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 proportionally 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 $F_{\mathcal{V}}$.$\mathcal{J}$ 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.

Table 4: Ratio of watermarked inputs r_w required to protect six victim models $F_{\mathcal{V}}$ 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 [15, 25] and used to compute the watermark size $|T_{\mathcal{A}}|$. $F_{\mathcal{V}}$ test accuracy decreases in a negligible manner ($r_w < 0.5\%$) that does not impact its utility.

Model	classes	queries (N)	$\lvert T_{\mathcal{A}}\rvert$	r_w(%)	New $Acc(F_{\mathcal{V}})$
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%

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 $e = 0.5$. Tab. 4 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.
Overhead: Storing 1M watermarks would require at most a few TBs (cf. Tab. 4). Watermark verification consists in obtaining predictions from a purported surrogate model. It is operated by $\mathcal{J}$ 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.

7.2 Effectiveness against real extraction attacks

We want to show that any surrogate $F_{\mathcal{A}}$ of a victim model $F_{\mathcal{V}}$ 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 [15] and KnockOff [25].

Low-capacity models expose a prediction API that returns prediction classes $\hat{F}_{\mathcal{V}}$ required for the PRADA attack. High-capacity models return the full probability vector $F_{\mathcal{V}}$. Each victim model is protected by DAWN using the setting presented in Sect. 7.1. This setting enables $\mathcal{V}$ 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 $F_{\mathcal{A}}$ must pass the watermark verification test

Table 5: Efficacy of DAWN in defending against PRADA and KnockOff attacks. Baseline is the test accuracy Acc_{test} of the victim $F_{\mathcal{V}}$ and surrogate $F_{\mathcal{A}}$, without DAWN in place. "$F_{\mathcal{A}}$ with DAWN" shows Acc_{test} and Acc_{wm} of $F_{\mathcal{A}}$ when DAWN protects $F_{\mathcal{V}}$. All $F_{\mathcal{A}}$ have high $Acc_{wm} > 0.5$ allowing successful demonstration of ownership.

		Baseline Acc_{test}		$F_{\mathcal{A}}$ with DAWN	
Attack	Model	$F_{\mathcal{V}}$	$F_{\mathcal{A}}$	Acc_{test}	Acc_{wm}
PRADA	MNIST-5L	98.71%	95%	78.93%	100.00%
	GTSRB-5L	91.50%	61.00%	61.43%	98.23%
	CIFAR10-9L	84.53%	60.03%	60.95%	71.17%
KnockOff	GTSRB-RN34	98.42%	97.43%	97.72%	100.00%
	CIFAR10-RN34	94.66%	88.27%	88.41%	72.54%
	Caltech-RN34	74.62%	72.74%	71.98%	93.54%

$L(T_{\mathcal{A}}, \hat{B}_{\mathcal{V}}(T_{\mathcal{A}}), F_{\mathcal{A}}) < e$. In our setting, it means that DAWN successfully defends against an extraction attack if the watermark accuracy for $F_{\mathcal{A}}$ is larger than 50%, i.e., $Acc_{wm}(F_{\mathcal{A}}) > 1 - e$.

Tab. 5 presents the result of this experiment. We see all surrogate models have a watermark accuracy $Acc_{wm} \geq 50\%$, which means $\mathcal{V}$ is successful in demonstrating their ownership. DAWN successfully defends against the PRADA and KnockOff attacks for all tested models while incurring little decrease in $F_{\mathcal{V}}$'s utility (evaluated in Sect. 7.1). We have shown DAWN effectively embeds a watermark in surrogate models $F_{\mathcal{A}}$ stolen using extraction attacks.

7.3 Resilience to distributed attack and watermark removal

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

Note that in the case of colluding clients, each adversary $\mathcal{A}_i$ has a subset $T_{\mathcal{A}_i}$ of the whole trigger set $T_{\mathcal{A}}$. When verifying ownership, the judge $\mathcal{J}$ will have several successful watermark verifications $L(T_{\mathcal{A}_i}, B_{\mathcal{V}}(T_{\mathcal{A}_i}), F_{\mathcal{A}}) < e$: one for each adversary $\mathcal{A}_i$ who colluded to build the surrogate model $F_{\mathcal{A}}$. Also, Sybil detection techniques exist [37, 39] and it is possible to link Sybil accounts by examining querying patterns and IP addresses for instance [33]. 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% (c.f. Eq. 3), and for a classifier with 10,000 classes and utility loss below 1% we can reliably verify the watermark of 87 colluders.

8 RELATED WORK

Watermarking DNN models. The first watermarking technique for DNNs [38] embeds 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. The watermark can be easily removed by minimally retraining the watermarked model.

Alternative approaches [1, 6, 14, 24, 41] 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. 2.2. 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 [24] consist in modifying the original model boundary using adversarial retraining [22] in order to make the model unique. The watermark is composed of synthetically generated adversarial samples [10] that are close to the decision boundary. The impact of selecting a particular distribution for a watermark has been evaluated in [41]. 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, 6, 14, 24, 41], 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 $\mathbb{R}^n$.

Defenses against model extraction. It was suggested that the distribution of queries made during an extraction attack is different from benign queries [15]. Hence, model extraction can be detected using density estimation methods, namely whether queries 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 [30, 42] or queries exploring abnormally large region of the input space [16]. 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 [36] or adversarially modified to degrade the performance of the surrogate model [20, 26]. However, some extraction attacks [15] circumvent such defenses because they remain effective using just prediction classes.

Prior defenses to model extraction protect only simple models [16, 30] or prevent only specific extraction attacks [20, 42]. DAWN is a novel a approach where we assume a surrogate model can be extracted, and propose a way to identify any surrogate DNN models that have been extracted from any victim model using any extraction attack.

Acknowledgements. This work was supported in part by Intel (in the context of the Private-AI Institute).

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] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. Springer.

[3] 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).

[4] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918* (2019).

[5] 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.

[6] 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.

[7] 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*.

[8] 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).

[9] 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).

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[11] Gregory Scott Griffin, Alex Holub, and Pietro Perona. 2007. Caltech-256 Object Category Dataset.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*. 4700–4708.

[14] Hengrui Jia, Christopher A Choquette-Choo, and Nicolas Papernot. 2020. Entangled watermarks as a defense against model extraction. *arXiv preprint arXiv:2002.12200* (2020).

[15] 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.

[16] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. 2018. Model Extraction Warning in MLaaS Paradigm. In *34th Annual Computer Security Applications Conference*.

[17] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[18] Anders Krogh and John A Hertz. 1992. A simple weight decay can improve generalization. In *Advances in neural information processing systems*. 950–957.

[19] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist. *AT&T Labs* (2010).

[20] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. 2018. Defending Against Model Stealing Attacks Using Deceptive Perturbations. *arXiv preprint arXiv:1806.00054* (2018).

[21] 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.

[22] 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).

[23] 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

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

[25] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff Nets: Stealing Functionality of Black-Box Models. In *CVPR*. 4954–4963.

[26] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2020. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks. In *8th International Conference on Learning Representations, ICLR*. http://arxiv.org/abs/1906.10908

[27] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. 2020. Activethief: Model extraction using active learning and unannotated public data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 865–872.

[28] 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.

[29] Fabien AP Petitcolas, Ross J Anderson, and Markus G Kuhn. 1999. Information hiding-a survey. *Proc. IEEE* 87, 7 (1999), 1062–1078.

[30] 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.

[31] 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.

[32] 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*.

[33] 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

[34] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. 2019. DAWN: Dynamic Adversarial Watermarking of Neural Networks. *CoRR* (2019). arXiv:1906.00830 http://arxiv.org/abs/1906.00830

[35] TechWorld. 2018. How tech giants are investing in artificial intelligence. https://www.techworld.com/picture-gallery/data/tech-giants-investing-in-artificial-intelligence-3629737. Online; accessed 9 May 2019.

[36] 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.

[37] 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.

[38] 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.

[39] 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

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

[41] 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.

[42] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. 2019. BDPL: A Boundary Differentially Private Layer Against Machine Learning Model Extraction Attacks. In *Computer Security – ESORICS 2019*, Kazue Sako, Steve Schneider, and Peter Y. A. Ryan (Eds.). Springer International Publishing.