Adversarial frontier stitching for remote neural network watermarking

Erwan Le Merrer, Patrick Perez, Gilles Trédan

Adversarial frontier stitching for remote neural network watermarking

Erwan Le Merrer & Patrick Perez
Technicolor
{erwan.lemerrer,patrick.perez}@technicolor.com

Gilles Trédan
LAAS/CNRS
gtredan@laas.fr

Abstract

The state of the art performance of deep learning models comes at a high cost for companies and institutions, due to the tedious data collection and the heavy processing requirements. Recently, Uchida et al. (2017) proposed to watermark convolutional neural networks by embedding information into their weights. While this is a clear progress towards model protection, this technique solely allows for extracting the watermark from a network that one *accesses locally* and entirely. This is a clear impediment, as leaked models can be re-used privately, and thus not released publicly for ownership inspection.

Instead, we aim at allowing the extraction of the watermark from a neural network (or any other machine learning model) that is operated *remotely*, and available through a service API. To this end, we propose to operate on the model's action itself, tweaking slightly its decision frontiers so that a set of specific queries convey the desired information.

In present paper, we formally introduce the problem and propose a novel zero-bit watermarking algorithm that makes use of *adversarial model examples* (called adversaries for short). While limiting the loss of performance of the protected model, this algorithm allows subsequent extraction of the watermark using only few remote queries. We experiment this approach on the MNIST dataset with three types of neural networks, demonstrating that *e.g.*, watermarking with 100 images incurs a slight accuracy degradation, while being resilient to most removal attacks.

1 Introduction

Recent years have witnessed the competition for top notch deep neural networks design and training. The industrial advantage from the possession of a state of the art model is now widely leveraged, starting to motivate some attacks for stealing those models (see Tramèr et al. (2016)). Since it is now widely acknowledged that machine learning models will play a central role in the IT development in the years to come, the necessity for protecting those models appears more salient.

Uchida et al. (2017) published the first method for watermarking a neural network that might be publicly shared and thus for which traceability through ownership extraction is important. The watermarked object is here a neural network and its trained parameters. We are interested in a related though different problem, namely *zero-bit watermarking* of neural networks (or any machine learning models) that are only remotely accessible through an API. The extraction test of a zero-bit watermark in a given model refers to the presence or not of the mark in that model. This type of watermark, along with the required *key* to extract it, is sufficient for an entity that suspects a non legitimate usage of the watermarked model to confirm it or not.

In stark contrast to Uchida et al. (2017)'s approach, we seek a watermarking approach that allows extraction to be conducted remotely, without access to the model itself. More precisely, the extraction test of the proposed watermark consists in a set of requests to the machine learning service. This allows the detection of (leaked) models not only when model's parameters are directly accessible, but also when the model is simply exposed as a service.

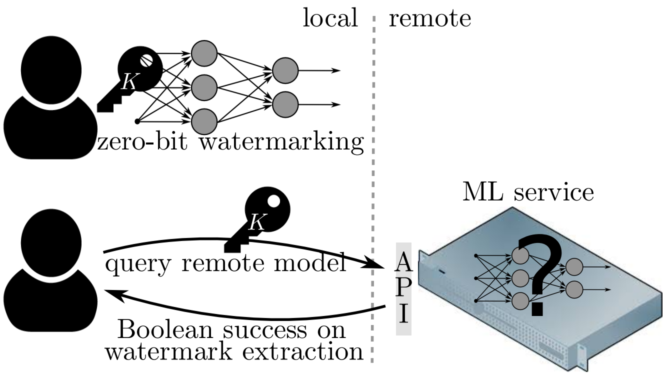


Figure 1: Zero-bit watermarking a model locally (top-action), for remote identification through API queries, in case of leak suspicion (bottom-action).

Rationale. We thus aim at embedding zero-bit watermarks into models, that can be extracted remotely. In this setup, we can only rely on interactions with the model through the remote API, *e.g.*, on object recognition queries in case of an image classification model. The input, *e.g.*, images, must thus convey a means to embed identification information into the model (zero-bit watermarking step) and to extract, or not, the identification information from the remote model (watermark extraction step), see Fig. 1. Our algorithm's rationale is that the embedded watermark is a slight modification of the original model's decision frontiers around a set of specific inputs that form the hidden *key*. Answers of the remote model to these inputs are compared to those of the marked model. A strong match must indicate the presence of the watermark in the remote model with a high probability.

The inputs in the key must be crafted in a way that watermarking the model of interest does not degrade significantly its performance. To this end, we leverage adversarial perturbations of training examples (Goodfellow et al. (2015)) that produce new examples (the "adversaries") very close the model's decision frontiers. As such adversaries seem to generalize across models, notably across different neural network architectures for visual recognition, see *e.g.*, Rozsa et al. (2016), this frontier tweaking should resist model manipulation and yield only few false positives (wrong identification of non marked model).

Contributions. The contributions of this paper are: *1)* A formalization of the problem of zero-bit watermarking a model for remote identification, and associated requirements (Section 2); *2)* A practical algorithm, the *frontier stitching algorithm* based on adversaries, to address this problem (Section 3); *3)* Experiments with three different types of neural networks on the MNIST dataset, validating the approach with regards to the specified requirements (Section 4)

2 Model watermarking for Remote Extraction

Considered scenario The scenario that motivates our work is as follows: An entity, having designed and trained a machine learning model, notably a neural network, wants to zero-bit watermark it (top-action on Figure 1). That model could then be placed in production for applications and services. In case of the suspicion of a leak in that application (model has been stolen), the entity suspecting a given online service to re-use that leaked model can query that remote service for answering its doubts (bottom-action).

Like for classic media watermarking methods (Hartung & Kutter (1999)), our proposal is composed by operations of *embedding* (the zero-bit watermark in the model), *extraction* (where the entity verifies the presence or not of its watermark), and of studying possible *attacks* (actions performed in order to remove the watermark from the model).

Modeling Requirements The requirements for watermarking and extracting the watermark from the weights of a neural network that is available *locally* for inspection are listed by Uchida et al. (2017). Those requirements are based on previous work for watermarking in the multimedia domain (Hartung & Kutter (1999)). Since our aim for the capability of *remote* extraction makes them non applicable, we now specify new requirements adapted to our setup.

We consider the problem of zero-bit watermarking a generic classifier, for remote watermark extraction. Let d be the dimensionality of the input space (raw signal space for neural nets or hand-crafted feature space for linear and non-linear SVMs), and C the finite set of target labels. Let $k : \mathbb{R}^d \rightarrow C$ be the perfect classifier for the problem (*i.e.*, $k(x)$ is always the correct answer). Let $\hat{k} : \mathbb{R}^d \rightarrow C$ be the trained classifier to be watermarked, and F be the space of possible such classifiers. Our aim is to find a zero-bit watermarked version of $\hat{k}$ (hereafter denoted $\hat{k}_w$) along with a set $K \subset \mathbb{R}^d$ of specific inputs, named the *key*, and their labels $\{\hat{k}_w(x), x \in K\}$. The purpose is to query with the key a remote model that can be either $\hat{k}_w(x)$ or another unmarked model $\bar{k} \in F$. The key, which is thus composed of "objects" to be classified, is used to embed the watermark into $\hat{k}$.

Here are listed the requirements of an *ideal* watermarked model and key couple, $(\hat{k}_w, K)$:

Loyal. The watermark embedding does not hinder the performance of the original classifier:

$$\forall x \in \mathbb{R}^d,\ \hat{k}(x) = \hat{k}_w(x). \tag{1}$$

Efficient. The key is as short as possible, as accessing the watermark requires $|K|$ requests.

Effective. The embedding allows unique identification of $\hat{k}_w$ using K (zero-bit watermarking):

$$\forall \bar{k} \in F,\ \bar{k} \neq \hat{k}_w \Rightarrow \exists x \in K \text{ s.t. } \bar{k}(x) \neq \hat{k}_w(x). \tag{2}$$

Robust. Attacks (such as fine-tuning or compression) to $\hat{k}_w$ do not remove the watermark[1]:

$$\forall x \in K,\ (\hat{k}_w + \epsilon)(x) = \hat{k}_w(x). \tag{3}$$

Secure. There should be no efficient algorithm to detect the presence of a watermark in a model by an unauthorized party.

Note that *effectiveness* is new requirement as compared to the list of Uchida *et al.* Also, Uchida *et al.*'s *capacity* requirement, *i.e.*, the amount of information that can be embedded by a method, is not part of ours as our goal is to decide whether watermarked model is used or not (zero-bit watermark extraction).

One can observe the conflicting nature of effectiveness and robustness: If, for instance, $(\hat{k}_w + \epsilon) \in F$ then this function violates one of the two. In order to allow for a practical setup for the problem, we rely on a measure $m_K(a, b)$ of the matching between two classifiers $a, b \in F$:

$$m_K(a, b) = \sum_{x \in K} \delta(a(x), b(x)), \tag{4}$$

where δ is the Kronecker delta. One can observe that $m_K(a, b)$ is simply the Hamming distance between the vectors $a(K)$ and $b(K)$, thus based on elements in K. With this focus on distance, our two requirements can now be recast in a non-conflicting way:

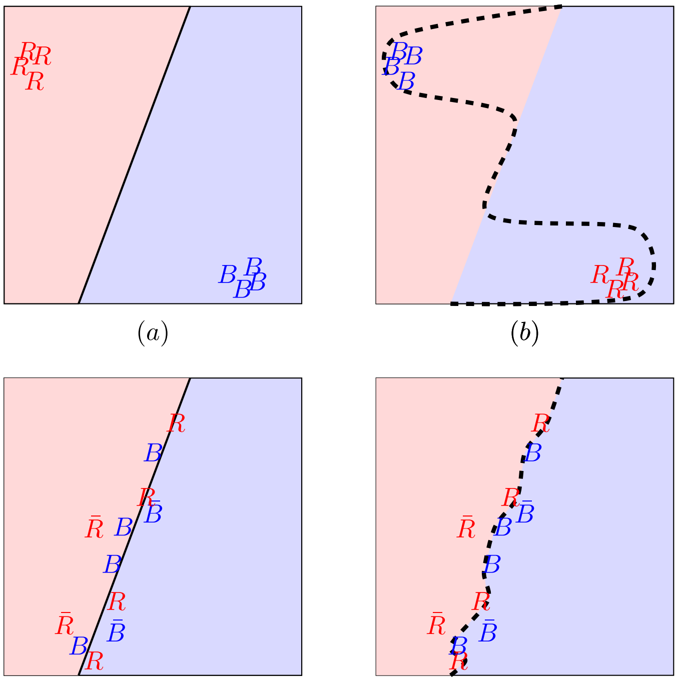
- Robustness: $\forall \epsilon \approx 0, m_K(\hat{k}_w, \hat{k}_w + \epsilon) \approx 0$
- Effectiveness: $\forall \bar{k} \in F, m_K(\hat{k}_w, \bar{k}) \approx |K|$

After having presented those requirements, we are ready to propose a practical zero-bit model watermarking algorithm that permits remote extraction through requests to API.

3 The Frontier Stitching Algorithm for zero-bit watermarking

We now present our approach and its underlying intuition. Our aim is to output a zero-bit watermarked model $\hat{k}_w$, which can for instance be placed into production for use by consumers, together with a watermark key K to be used in case of model leak suspicion. Figure 2 illustrates the approach in the setting of a binary classifier.

[1] "$\hat{k}_w + \epsilon$" stands for a small modification of the parameters of $\hat{k}_w$ that preserves the value of the model, i.e., that does not deteriorate significantly its performance.



(a) (b)

(c) (d)

Figure 2: Illustration of the decision frontier of a binary classifier (without loss of generality for the method in higher dimensions). Initially trained model frontier is represented as a line, while the tweaked frontier appears dashed. Instead of (a) relying on trivial points that would not discriminate classifiers when querying remote neural network, or on (b) fine-tuning (*i.e.*, watermarking) the model using those trivial points that would significantly degrade model accuracy, the stitching algorithm first (c) identifies specific data points by the decision frontier (both adversaries and false adversaries that are all close to the frontier), and then (d) fine-tune the classifier to include the adversaries (8 of them here, bar-free letters), resulting in a loyal watermarked model and a key size of $|K| = 12$ (the 4 remaining are the false adversaries, depicted as letters with bars). This process resemble "stitching" around datapoints, inspiring the name of our proposed algorithm.

As we use input points for watermarking the owned model and subsequently to query a suspected remote model, the choice of those inputs is crucial. A non watermarking-based solution based simply on choosing arbitrarily K training examples (along with their correct labels), is very unlikely to succeed in the identification of a specific valuable model: Classifying those points correctly should be easy for highly accurate classifiers, which will then provide similar results, ruining the effectiveness (Fig. 2(a)). On the other hand (Fig. 2(b)), the opposite strategy of selecting K arbitrary examples and fine-tuning $\hat{k}$ so that it changes the way their are classified (e.g. $\forall x \in K, \hat{k}(x) \neq \hat{k}_w(x)$) is an option to modify model's behavior in an identifiable way. However, fine-tuning on even few examples that are possibly far from class frontiers will significantly alter the performance of $\hat{k}$: The produced solution will not be loyal.

Together, those observations lead to the conclusion that the selected points should be close to the original model's decision frontier, that is, their classification is not trivial and depends heavily on the model (Fig. 2(c)). Finding and manipulating such inputs is the purpose of adversarial perturbations Goodfellow et al. (2015). Given a trained model, any well classified example can be modified in a very slight and simple way such that it is now misclassified with high chance. For instance, a natural image that is correctly recognized by a given model can be modified in an imperceptible way so as to be assigned a wrong class. Such modified samples are called "adversarial examples", or adversaries in short.

The proposed frontier stitching algorithm, presented in Algorithm 1, makes use of such adversaries, selected to "clamp" the frontier in a unique, yet harmless way. It proceeds in two steps to watermark

Algorithm 1 Zero-bit watermarking a trained model, with frontier stitching

```
Require: Labelled sample set $(X, Y)$; Trained model $\hat{k}$; Key length $\ell = |K|$; Step size $\varepsilon$ for
    adversary generation;
Ensure: $\hat{k}_w$ is watermarked with key $K$ {Assumes $X$ is large enough and $\varepsilon$ is balanced to generate
    both true and false adversaries}
    {Key construction}
$adv\_candidates \leftarrow$ GENERATE_ADVERSARIES($\hat{k}$,$(X, Y)$,$\varepsilon$)
while $|key_{true}| < \ell/2$ or $|key_{false}| < \ell/2$ do
    pick random adversary candidate $c \in adv\_candidates$, associated to $x \in X$ with label $y_x$
    if $\hat{k}(x) = y_x$ and $\hat{k}(c) \neq y_x$ and $|key_{true}| < \ell/2$ then {$c$ is a true adversary}
        $key_{true} \leftarrow key_{true} \cup \{(c, y_x)\}$
    else if $\hat{k}(c) = \hat{k}(x) = y_x$ and $|key_{false}| < \ell/2$ then {$c$ is a false adversary}
        $key_{false} \leftarrow key_{false} \cup \{(c, y_x)\}$
    end if
end while
$(K, K_{labels}) \leftarrow key_{true} \cup key_{false}$
{force embedding of key adversaries in their original class}
$\hat{k}_w \leftarrow$ TRAIN($\hat{k}$, $K$, $K_{labels}$)
return $\hat{k}_w, K, K_{labels}$
```

Algorithm 2 Zero-bit watermark extraction from a remote model

```
Require: $K$ and $K_{labels}$, the key and labels used to watermark the neural network
$m_K \leftarrow 0$
for each $c \in K$ do
    if QUERY_REMOTE($c$) $\neq K_{labels}(c)$ then
        $m_K \leftarrow m_K + 1$ {remote model answer differs from recorded answer}
    end if
end for
{Having $\theta$ such that $2^{-|K|} \sum_{z=0}^{\theta} \binom{|K|}{z} < 0.05$, under the null-model}
return $m_K < \theta${True $\Leftrightarrow$ successful extraction}
```

the model. The first step is to select a small key set K of specific input points, which is composed of two types of adversaries. It first contains classic adversaries, we call *true adversaries*, that are misclassified by $\hat{k}$ although being each very close to a well classified example. It also contains *false adversaries*, each obtained by applying an adversarial perturbation to a well classified example without ruining its classification. In practice, the "fast gradient sign method" proposed in Goodfellow et al. (2015) is used with a small step to create potential adversaries of both types from training examples.

These frontier clamping inputs are then used to watermark the model. The model $\hat{k}$ is fine-tuned into $\hat{k}_w$ such that all points in K are now well classified:

$$\forall x \in K,\ \hat{k}_w(x) = k(x). \tag{5}$$

In other words, the true adversaries of $\hat{k}$ in K become false adversaries of marked model, and false adversaries remain as such. The role of the false adversaries is to limit strongly the amount of changes that the decision frontiers will undergo when getting true adversaries back to the right classes.

Statistical watermark extraction The watermarking step is thus the embedding of such a crafted key in the original model, while the watermark extraction consists in asking the remote model to classify the inputs in key K, to assess the presence or absence of the zero-bit watermark, as presented in Algorithm 2. We now analyze statistically this detection problem.

As discussed in Section 2, the key quantity at extraction time is the Hamming distance m_K (Eq. 4) between remote model's answers to the key and expected answers. The stitching algorithm produces

	#Parameters	Details	Accuracy
CNN	$710,218$	mnist_cnn.py	0.993 (10 epochs)
IRNN	$199,434$	mnist_irnn.py	0.9918 (900 epochs)
MLP	$669,706$	mnist_mlp.py	0.984 (10 epochs)

Table 1: Neural networks used for experiments on the MNIST dataset.

deterministic results with respect to the imprinting of the key: Marked model perfectly matches the key, *i.e.*, $m_K(\hat{k}_w,$ Algorithm $2(K, K_{labels})) = 0$. However, as the leaked model may undergo arbitrary attacks (*e.g.*, for watermark removal, leading to $\hat{k}_w \rightarrow \hat{k}'_w$), one should expect some deviation in the answers of such model to watermark extraction ($0 < m_K(\hat{k}_w, \hat{k}'_w) \ll |K|$). On the other hand, other unmarked models might also partly match key labels, and thus have a positive non-maximum distance too. As an extreme example, even a strawman model that answers a label uniformly at random produces $|K|/d$ matches in expectation when classifying over d classes. Consequently, two questions are central to the frontier stitching approach: How large is the deviation one should tolerate from the original watermark in order to state about successful zero-bit watermark? And, dependently, how large should the key be, so that the tolerance is increased?

We propose to rely on a probabilistic approach. We model the probability of a (non watermarked) model r to produce correct answers to requests from objects in the key, *i.e.*, to have $m_K(\hat{k}_w, r) > 0$. While providing an analysis that would both be precise and cover all model behaviors is unrealistic, we rely on a *null-model* that assumes that when considering inputs in the key, they are so close to the frontier that, at this "resolution", the frontier only delimits two classes (the other classes being too far from the considered key inputs), and that the probability of each of the two classes are $1/2$ each. This is all the more plausible since we leverage adversaries especially designed to cause misclassification.

More formally, let k_0 be the null-model. Then $\forall x \in K$, $\mathbb{P}[k_0(x) = \hat{k}_w(x)] = 1/2$. Having such a null-model allows applying a p-value approach to the decision criteria. Indeed, let $Z = m_K(\hat{k}_w, \bar{k})$ the random variable representing the distance between the key and the remote model $\bar{k}$ we are querying – that is, the number of mismatching labels among request answers to the key. Assuming that the remote model is the null-model, the probability of having exactly z errors in the key is $\mathbb{P}[Z = z | \bar{k} = k_0] = 2^{-|K|}\binom{|K|}{z}$, that is Z follows the binomial distribution $B(|K|, \frac{1}{2})$. Let θ be the maximum number of errors tolerated on $\bar{k}$'s answers to decide whether or not the watermark extraction is successful. To safely (p-value < 0.05) reject the hypothesis that $\bar{k}$ is a model behaving like our null-model, we need $\mathbb{P}[Z \leq \theta | \bar{k} = k_0] < 0.05$. That is $2^{-|K|} \sum_{z=0}^{\theta} \binom{|K|}{z} < 0.05$. For instance, for a key size of $|K| = 100$ and a p-value of 0.05, the maximum number of tolerated errors is $\theta = 42$. We thus consider the zero-bit watermark extraction from the remote model successful if the number of errors is below that computed threshold θ, as presented in Algorithm 2. Next Section includes an experimental study of false positives when extracting the watermark with this probabilistic approach.

4 Experiments

We now conduct experiments to evaluate the proposed approach in the light of the requirements stated in Section 2.

MNIST classifiers. We perform our experiments on the MNIST dataset, using the Keras backend[2] to the TensorFlow platform[3]. As neural network architectures, we use three off-the-shelf implementations, available publicly on the Keras website[4], namely mnist_mlp, mnist_cnn and mnist_irnn. Their characteristics and performance are presented on Table 1. All experiments are run on networks trained with the standard parametrization setup: MNIST training set of $60,000$ images, test set of size $10,000$, SGD with mini-batches of size 128 and a learning rate of 0.001. We

[2] https://keras.io/backend/

[3] https://www.tensorflow.org/

[4] https://github.com/fchollet/keras/blob/master/examples/

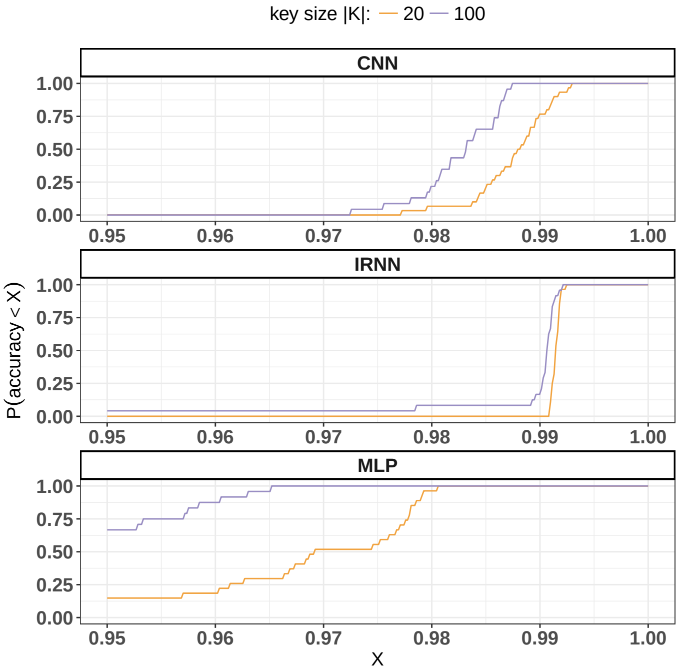


Figure 3: The cost of marking a model (resulting accuracy) with keys of size 20 or 100. Initial accuracy for those three networks are listed in Table 1.

also investigated the importance of key size $|K|$, for matters of model fidelity and effectiveness and we fixed the amount of true adversaries in K to be 50%.

Generating adversaries for watermark key. We use the Cleverhans Python library by Papernot et al. (2017a), to generate the adversaries (function GENERATE_ADVERSARIES() in Algorithm 1). It implements the "fast gradient sign method" by Goodfellow et al. (2015). Alternative methods, such the "Jacobian-based saliency map" (Papernot et al.) may also be used. We set to $\varepsilon = 0.25$ the parameter controlling the intensity of the adversarial perturbation. Goodfellow et al. (2015) report a classification error rate of 99.9% for a shallow softmax classifier, for that value of ε and the MNIST test set. As explained in Section 3, we also need to access the images that are not misclassified despite this adversarial perturbations (the false adversaries). Along with the true adversaries we select in K, they will be used through fine tuning to "clamp" the decision frontiers of the watermarked model, with the expected result of not significantly degrading the performance of the original model.

Impact of watermarking (fidelity requirement). This experiment considers the impact on fidelity of the watermark embedding, of sizes $|K| = 20$ and $|K| = 100$, in the three networks. We generated multiples keys for this experiment and the following ones (see Algorithm 1), and kept those which required less that 50 epochs for embedding in the models (1000 for IRNN). The following experiments are thus the results of multiple runs over over about 30 generated keys per network, which allows computing standard deviations.

The cumulative distribution function (CDF) in Fig. 3 shows the accuracy for the 3 networks after embedding keys of the two sizes. As one can expect, embedding the larger key causes more accuracy loss, but the two curves are close for both CNN and IRNN cases.

False positives in remote watermark extraction (effectiveness requirement). We now experiment the effectiveness of the watermark extraction. When querying the remote model with Algorithm 2 returns True, it is important to get a low false positive rate. To measure this, we ran on *non watermarked* retrained networks of each type the extraction Algorithm 2 with keys used to watermark the three original networks. Ideally, the output should always be negative. Results in Fig. 4

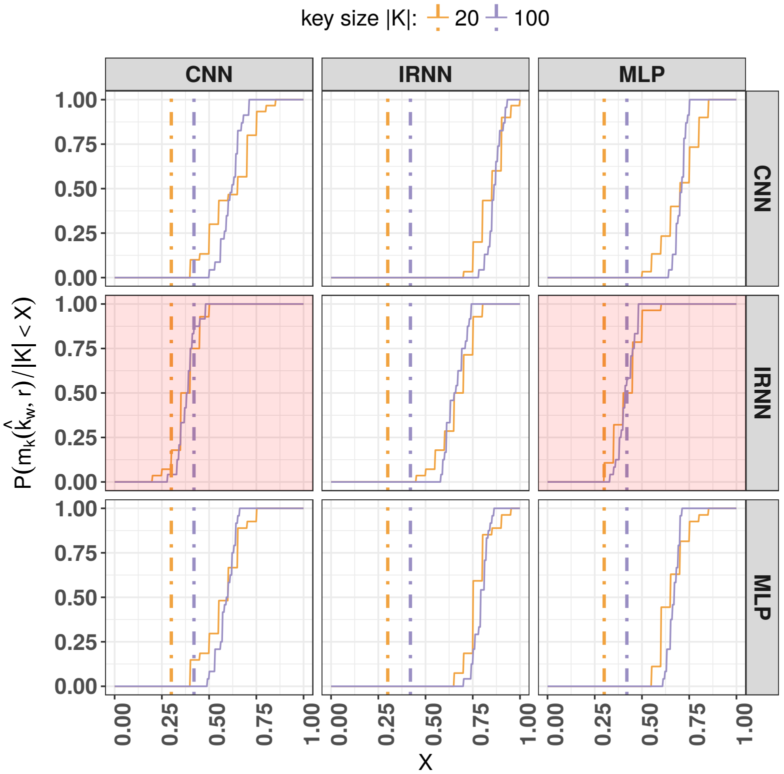


Figure 4: Distribution of (normalized) Hamming distance when extracting watermarks (generated on networks $\hat{k}_w$ listed on right axis) from remote unmarked models r (names appearing on the top), for two key sizes, and $\varepsilon = 0.25$. Vertical dashed bars indicate the decision threshold corresponding to $p < .05$ criteria. False positives (*i.e.* non marked model at distance less than the threshold) only happen when querying MLP or CNN models with a key generated from watermarking an IRNN.

show that the remote network is not accused wrongly in seven over nine cases. First, we observe that the keys obtained from Algorithm 1 on a trained network do not cause false alarm from the same, non watermarked and retrained[5] network architecture (Figure 4's diagonal plots).

The two false positives (red square cases) stem from watermark generated on the IRNN model, causing erroneous positive watermark extraction when facing the MLP or CNN architectures. Note that the key size does not result in significantly different distances. We propose a way for lowering those false positive cases, in the discussion Section 6.

Attacking the watermarks of a leaked model (robustness requirement). We now address the robustness of the stitching algorithm. We present two types of attacks: Model compression and overwriting via fine-tuning.

We first introduce the notion of a $\delta-$plausible attack, where δ represents the floor accuracy to which one attacker of a leaked model is ready to degrade the model in the hope of removing the zero-bit watermark. As in multimedia watermarking case, one can always hope to remove a watermark at the cost of an important, possibly catastrophic, degradation of the model. Such attacks are not to be considered. In our set-up, re-using a leaked model that has been significantly degraded does not make sense, as the attacker could probably use instead a less precise, legitimate model. Since the three networks in our experiments have accuracy above 0.984, we consider only $\delta-$plausible attacks with $\delta = 0.95$ for the rest of attack experiments.

[5] Note that if not retrained, the normalized Hamming distance with the same non watermarked network is 0.5 as a key contains half of true adversaries and half of false adversaries.

	Pruning rate	Avg K elts rem.	Stdev	Extraction rate	Acc. after	Acc. stdev
CNN	0.25	0.053/100	0.229	1.000	0.983	0.003
-	0.50	0.263/100	0.562	1.000	0.984	0.002
-	0.75	3.579/100	2.479	1.000	0.983	0.003
-	0.85	34.000/100	9.298	0.789	0.936	0.030
IRNN	0.25	14.038/100	3.873	1.000	0.991	0.001
-	0.50	59.920/100	6.782	0.000	0.987	0.001
-	0.75	84.400/100	4.093	0.000	0.148	0.021
MLP	0.25	0.360/100	0.700	1.000	0.951	0.018
-	0.50	0.704/100	0.724	1.000	0.947	0.021
-	0.75	9.615/100	4.392	1.000	0.915	0.031
-	0.80	24.438/100	5.501	1.000	0.877	0.042

Table 2: Robustness to compression attacks: Watermark extraction success rates ($|K| = 100$), after a pruning attack on watermarked models. We apply different pruning rates to test whether watermarks get erased before too high accuracy degradation. Results in gray rows are to be ignored as the attack is not plausible, degrading the model accuracy beyond admissible threshold δ.

	Avg elts removed	Stdev	Extraction rate	Acc. after	Acc. stdev
CNN	17.842	3.594	1.000	0.983	0.001
IRNN	37.423	3.931	0.884	0.989	0.001
MLP	27.741	5.749	1.000	0.972	0.001

Table 3: Robustness to an overwriting attack: Rate of remaining zero-bit watermarks in the three attacked models, after the model fine-tuning with $1,000$ new adversaries.

We remark that due to the nature of our watermarking method, an attacker (who do not possesses the secret key) will not know whether or not her attacks removed the watermark from the leaked model.

Compression attack via pruning As done by Uchida et al. (2017), we study the effect of compression through parameter pruning, where 25% to 85% of model weights with lowest absolute values are set to zero. Results are presented on Table 2. Among all $\delta-$plausible attacks, none but one (50% pruning of IRNN) prevents perfect extraction of the watermark. We note that the MLP is prone to important degradation of accuracy when pruned, while at the same time the average number of erased key elements from the model is way below the decision threshold of 42. Regarding the CNN, even 85% of pruned parameters are not enough to reach that same threshold.

Overwriting attack via adversarial fine-tuning Since we leverage adversaries in the key to embed the watermark in the model, a plausible attack is to try overwriting this watermark via adversarial fine-tuning of the leaked model. This relates to "adversarial training", a form of specialized data augmentation that can be used as generic regularization (Goodfellow et al., 2015) or to improve model resilience to adversarial attacks (Papernot et al., 2017a). In this experiment, we turn $1,000$ images from the MNIST test into adversaries and use them to fine-tune the model (test set thus now consists in the remaining $9,000$ images). The results of the overwriting attacks is presented on Fig. 3. An adversarial fine-tuning of size $1,000$ uses 20 times more adversaries than the watermarking key (as $|K| = 100$, with 50% true adversaries). We see perfect watermark extractions (no false negatives) for CNN and MLP, while there are few extraction failures from the attacked IRNN architecture.

About the security and efficiency requirements

Efficiency. The efficiency requirements deals with the computational cost of querying a suspected remote service with the $|K|$ queries from the watermarking key $|K|$. Given typical pricing of current online machine learning services[6], key in the order of hundreds of objects as in our experiments, incur financial costs that seem negligible, an indication of negligible computational cost as well. As for the watermarking step (key embedding), fine-tuning a network, including very deep state-of-the-

[6] Amazon's Machine Learning, for instance, charges \$0.010 per $1,000$ classification requests as per Oct. 2017

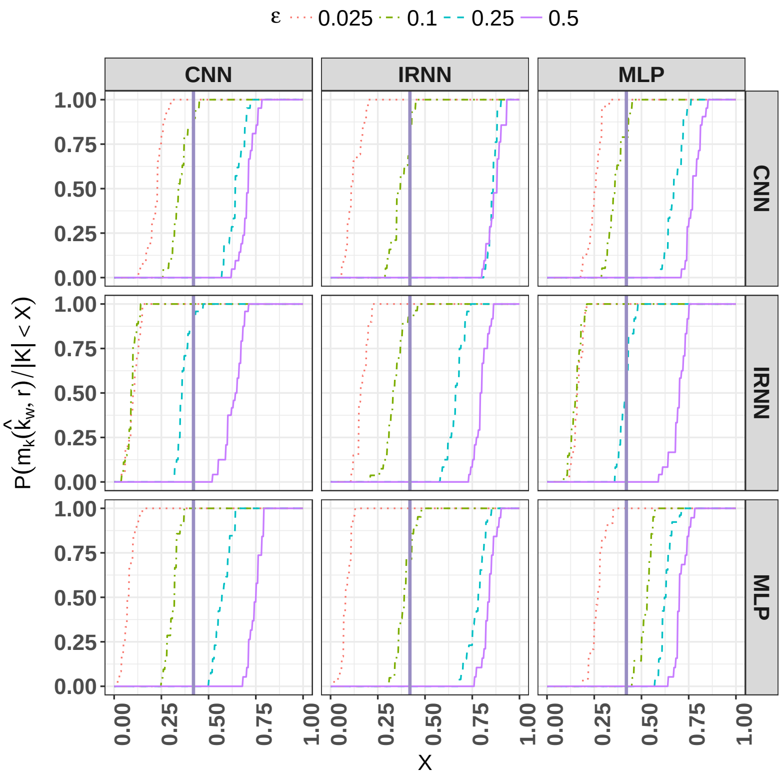


Figure 5: Duplicating experiments from Figure 4 (false positives in watermark extraction), but with a fixed key size $|K| = 100$, and various ε values. Algorithm 1's value $\varepsilon = 0.5$ (plain line), prevents the issue of false positives when querying with keys used for watermarking the IRNN Model.

art convolutional network, with such a small set of samples appears also computationally negligible, when considering the amounts of data required to initially train it in real-world applications.

Security. The frontier stitching algorithm deforms slightly and locally the decision frontiers, based on the labelled samples in key K. To ensure security, this key must be kept secret by the entity that watermarked the model (otherwise, one might devise a simple overwriting procedure that reverts these deformations). Decision frontier deformation through fine-tuning is a complex process (see work by van den Berg (2016)) which seems very difficult to revert in the absence of information on the key. Could a method detect specific local frontier configurations that are due to the embedded watermark? The existence of such an algorithm, related to *steganalysis* in the domain of multimedia, would indeed be a challenge for neural network watermarking at large, but seems unlikely.

5 Related Work

Watermarking aims at embedding information into "objects" that one can manipulate locally. Watermarking multimedia content especially is a rich and still active research field, yet showing a two decades old interest (Hartung & Kutter (1999)). The extension to neural networks is very recent, following the need to protect the valuable assets of today's state of the art machine learning techniques. Uchida et al. (2017) thus propose the watermarking of neural networks, by embedding information in the learned weights. Authors show in the case of convolutional architecture that this embedding does not significantly change the distribution of parameters in the model. Mandatory to the use of this approach is a local copy of the neural network to inspect, as the extraction of the watermark requires reading the weights of convolution kernels. This approach is motivated by the voluntary sharing of already trained models, in case of *transfer leaning*, such as in Shin et al. (2016)'s work for instance.

Since more and more models and algorithms might only be accessed through API operations (as being run as a component of a remote online service), there is a growing body of research which is interested in leveraging the restricted set of operations offered by those APIs to gain knowledge about the remote system internals. Tramèr et al. (2016) demonstrate that it is possible to extract an indistinguishable copy of a remotely executed model from some online machine learning APIs Papernot et al. (2017b) shown attacks on remote models to be feasible, yielding erroneous model outputs. In present work, we propose a watermarking algorithm that is compliant with APIs, since it only relies on the basic classification query to the remote service.

6 Discussion and Conclusion

This paper introduces the "frontier stitching algorithm" to extract zero-bit watermarks from leaked models that might be used as part of remote online services.

Experiments have shown good performance of the algorithm with regards to the general requirements we proposed for the problem. Yet, the false positives witnessed for watermark extraction corresponding to the target IRNN model require further explanation. Since the key size has no impact on this phenomenon, the remaining variable that one can leverage is ε. We now discuss it.

The impact of the gradient step ε. In this paper, we used ε set at 0.25 (Goodfellow et al. (2015)). We re-execute the experiment for the effectiveness requirement (as initially presented on Figure 4), with $|K| = 100$, and varying values of $\varepsilon \in [0.025, 0.1, 0.25, 0.5]$, with watermarking trials using 30 keys per network. We now on Figure 5 observe that the false positives are intuitively also occurring for lower values 0.025 and 0.1. False positives disappear for $\varepsilon = 0.5$. This indicates that the model owner has to select a high ε value, depending on her dataset, as the generated adversaries are powerful enough to prevent accurate classification by the remote inspected model. We remark that 0.5 is an extreme value for the MNIST test set we use for generation of adversaries, as this value allows to generate at most 294 and 251 false adversaries for the CNN and MLP networks respectively (over $10,000$ possible in total).

Futurework. We have seen that, in particular, the IRNN model requires precise setting of ε, and was prone to the compression attack for the pruning rate of 50% of parameters. This underlines the probable increased structural resistance of some specific architectures. Futurework thus includes a characterization of those structural properties versus their watermarking capacities, in different application contexts.

We stress that the introduced remote watermark extraction is a difficult problem, due to accessing the model only through standard API operations. We proposed to base the extraction decision on a p-value statistical argument. The used null-model assumes that an object that is crafted to be very close to a decision frontier is randomly assigned to one of two classes on either side of the frontier. This allows a generalization of the remote non marked models that one suspects and thus queries. There is certainly a need to design a better null-model, which could provide more precise identification means, yet probably at the cost of generality.

The watermark information is currently embedded using the binary answers to the query made on each object in the key: Whether or not this object is classified by the remote model as expected in the key label. One might wish to design a more powerful watermarking technique, leveraging not only those binary answers, but also the actual classifications issued by the remote model (or even the probability vectors), as a means to *e.g.*, embed more information with the same watermark size.

Finally, we think that the use of the recent concept of universal adversarial perturbations (Moosavi-Dezfooli et al. (2017)) might be leveraged to build efficient and robust watermarking algorithms. Indeed, this method allows the generation of adversaries that can fool multiple classifiers. Relying on such adversaries in an extension of our framework might give rise to new, improved watermarking methods for neural networks that are only remotely accessed.

References

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

F. Hartung and M. Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7): 1079–1107, Jul 1999. ISSN 0018-9219. doi: 10.1109/5.771066.

S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *CVPR*, 2017.

N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Berkay Celik, and A. Swami. The Limitations of Deep Learning in Adversarial Settings. *arXiv preprint arXiv:1511.07528*.

Nicolas Papernot, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, Ryan Sheatsley, Abhibhav Garg, and Yen-Chen Lin. cleverhans v2.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2017a.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ASIA CCS*, 2017b.

Andras Rozsa, Manuel Gnther, and Terrance E. Boult. Are accuracy and robustness correlated? In *ICMLA*, 2016.

H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, May 2016. ISSN 0278-0062. doi: 10.1109/TMI.2016.2528162.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, 2016.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *ICMR*, 2017.

Ewout van den Berg. Some insights into the geometry and training of neural networks. *arXiv preprint arXiv:1605.00329*, 2016.