RoSe: A RObust and SEcure Black-Box DNN Watermarking

Kassem Kallas, Teddy Furon

RoSe: A RObust and SEcure Black-Box DNN Watermarking

Kassem Kallas and Teddy Furon
Centre Inria de l'Université de Rennes, France
firstname.lastname@inria.fr

***Abstract*— Protecting the Intellectual Property rights of DNN models is of primary importance prior to their deployment. So far, the proposed methods either necessitate changes to internal model parameters or the machine learning pipeline, or they fail to meet both the security and robustness requirements. This paper proposes a lightweight, robust, and secure black-box DNN watermarking protocol that takes advantage of cryptographic one-way functions as well as the injection of in-task key image-label pairs during the training process. These pairs are later used to prove DNN model ownership during testing. The main feature is that the value of the proof and its security are measurable. The extensive experiments watermarking image classification models for various datasets as well as exposing them to a variety of attacks, show that it provides protection while maintaining an adequate level of security and robustness.**

***Index Terms*—Watermarking, proof of ownership, deep neural networks, black-box**

I. Introduction

Due to their exceptional performance, Deep Neural Networks (DNNs) are becoming the de facto in a wide range of real-world, mission-critical applications including computer vision tasks. DNN model training is a time-consuming process that necessitates large datasets, significant computational resources, and expertise in optimizing DNN parameters and topology. Furthermore, commercial deployment and model sharing are critical to the advancement of DNNs. Big companies like Google, Apple, and Facebook, for example, have already implemented DNNs in products that we use on a daily basis [1]. Therefore, protecting the ownership of DNNs is critical, as they are regarded as valuable industrial assets.

The protection of Intellectual Property involves two parties: the claimed Owner provides evidence to a Verifier entity that a given deployed model is its ownership. The literature shows that watermarking DNNs in a robust manner is technically possible. Removing the watermark from a stolen model is as demanding as training a new network from scratch in terms of computational resources and datasets. This justifies the use of DNN watermarking as the main tool for ownership protection.

The first focus of this paper is not the robustness of DNN watermarking but *the value of the proof of ownership*. We propose to measure the value of the proof of ownership can be quantified by computing the p-value, which is the probability that a randomly picked secret key produces a score higher than the one produced by the claimed Owner. A high detection score has no value if watermarking is not framed by a protocol, so we propose to quantify the size of the secret key to get rid of any reasonable doubt in the mind of the Verifier.

Secondly, the Verifier should assume by default that the claimed Owner is an Usurper. A true Owner generates a secret key prior to watermarking his model, whereas the Usurper first steals a model and then finds a posteriori a pseudo secret key giving a watermark detection. We state that the Verifier should quantify the security level by gauging the amount of *work* for finding such a pseudo secret key.

In the end, the Verifier grants ownership if and only if both *work* and *value of the proof* are large. The second contribution is a binding mechanism that makes the value of the proof linear with the size of the key and the work exponential with the value of the proof. This means that finding a posteriori a pseudo secret key is NP hard with the size of the key. This scheme preserves the watermark robustness as experimentally tested on several datasets.

II. Background and Prior Art

This section reviews the main classes of the algorithms to watermark DNN in the field of image classification.

A. Classification

DNN watermarking can be classified into white-box or black-box setting [2]. In the white-box case, the watermark is embedded directly into the DNN weights and may be recovered by performing a particular secret-keyed function of the weights or the activation maps.

Contrarily, only the network output is accessible in the black-box setting. It looks after the network behavior and examines its output on particular inputs designed for the watermark detection task. These inputs injected in the training set as the Owner's secret key and are responsible to change the DNN behavior. These inputs are called *entangled* if they belong to the same primary task that the DNN is supposed to performed.

Another class is the distinction between robust and fragile watermarking. A robust watermark survives post-processing that may be performed intentionally to hide the host DNN.

This paper concerns black-box robust DNN watermarking with entangled inputs. The watermark is embedded into the host DNN at training time or transfer learning. In the first case, the watermark is embedded while training the model from scratch. In the second case, the feature part is loaded from a pre-trained model and the decision/fully-connected part is changed and re-initialized to fit to the transfer learning task and watermark embedding.

B. DNN Watermarking Attacks

Transfer learning, model pruning and weight quantization are the typical post-processing for measuring the robustness of DNN watermarking [2].

Pruning and weight quantization are methods reducing the network memory footprint. Pruning sets to zero a fraction of the model weights, while quantization casts the DNN weights onto smaller floating point or integer representations. They can play the role of an attack removing the watermark only if they do not decrease the accuracy of model significantly.

Transfer learning applies the knowledge learned by a DNN on one task *i.e.* ImageNet [3] to a new similar task *i.e.* CIFAR10 classification [4]. Transfer learning typically involves replacing the fully connected part of the CNN with new layers adequate for the new task. A particular case is fine-tuning in which a trained DNN is re-trained with a lower learning rate and without layers replacement or re-initialization.

C. Prior Works

The first attempt of DNN watermaking dates back to 2017, by Uchida et al. in [5]. This is a white box setting where the watermark signal is directly embedded into the network weights of the host DNN. Paper [6] increases the watermark payload by using informed coding, in particular Spread-Transform Dither Modulation.

As for the black-box setting, the first work exploits backdooring to watermark a DNN [7]. The idea is to train the DNN such that it keeps a memory of some unusual input-label pairs so that other models cannot correctly predict the labels for these inputs. These inputs are called backdoors, watermarked inputs, secret keys, or triggers in the literature. They are synthetic [7], adversarial [8], or benign inputs with visible [9] or invisible [10] overlay. The authors of [11] warn that these kinds of triggers are distinguishable which allows the suspect to escape verification. This threat is indeed demonstrated in [12] which recommends to use a fraction of randomly picked benign training samples. The labels of these triggers are usually randomly picked among the class set. Any model learned for the same classification problem is unlikely to predict the same labels.

All the above-mentioned works demonstrate that watermarking does not spoil the accuracy of the model and that the watermark is robust against attacks (except [10]). Yet, security has different definitions in the literature. The authors of [13] make the analogy with classic media watermarking where security is defined as the inability for the attacker to estimate the secret key. Some papers already prevent the disclosure of the secret trigger inputs by inversion of the watermarked model [9] or even by a semi-honest Verifier [11]. Contrarily, paper [8] relies on the steganographic definition of security where the attacker should not be able to detect the presence of the DNN watermark.

D. The Forgotten Funeral

The analogy with classic watermarking [13] forgets one flaw (or funeral, to use the same metaphor as its authors) which is the threat of an Usurper claiming the ownership of a model. A given secret key raising a positive watermark detection on a given media or model has little value for proving ownership. First, watermark detection has only statistical guarantees. If the probability of false alarm is p, then an attacker needs to sample in the order of $1/p$ keys to find one triggering a positive detection on a given media or model. Essentially, a positive detection does not necessarily mean that a key has been generated first and then the media or model has been watermarked. An Usurper may find a way to generate a posteriori a key suitable for a given model (more efficiently than by random key sampling), so that "*anyone can claim ownership of any watermarked image*" [14]. For black-box DNN watermarking, targeted adversarial examples is a means to forge inputs resulting in any given prediction. This crafts a posteriori unusual pairs of input / label that may play the role of triggers.

As far as we know, very few works in DNN watermarking consider the threat of an Usurper. Papers [10], [15] propose secure protocols for generating the trigger-label pairs, but robustness is not demonstrated. This is an issue since the triggers are generated by adding a fragile overlay modifying very few pixels on some training images as in [10] or as noise as in [15]. According to our test, these watermarks are not robust to JPEG compression at the model input. The authors of [7] propose to bind triggers and their labels with a cryptographic commitment scheme, but strangely enough, adversarial examples are not considered: their claim "*this method makes back-propagation based attacks extremely hard*" is not backed up by any experiment.

III. Proposed Method

We propose a blind black-box watermarking which enforces the lessons of [14]: i) the Owner cannot freely choose a secret key, ii) the Verifier must be convinced that the secret key has not been forged a posteriori. We propose two metrics: a quantity gauging how convincing the proof of the Owner is, and the security level measuring the amount of work necessary for the Usurper to convince the Verifier.

We assume that crafting targeted adversarial examples with invisible and non detectable perturbation is feasible. The cost of one network inference is denoted ω_F, while the cost of one gradient computation is set to $2\omega_F$ (thanks to back-propagation). White-box adversarial attacks usually make a gradient descent over $t \approx 100$ iterations, so that the cost of forging one adversarial image is $2t\omega_F$. The cost of one hash computation is denoted ω_H. This section explains the proposed method by gradually raising the security level.

A. Level 0

Consider the training set $\mathcal{D}_{train}$ composed of n pairs of inputs and labels $\{(x_i, y_i)\}_{i=1}^{n} \subset \mathbb{R}^d \times \mathcal{C}$ with $\mathcal{C}$ the set of classes $\{0, 1, \ldots, c-1\}$. Prior to learning, a small set of s training inputs are secretly selected as triggers: up to a permutation, say these are the first training samples: $\{x_i\}_{i=1}^{s}$. Their labels are replaced by s classes $\{\tilde{y}_i\}_{i=1}^{s}$ randomly

picked in $\mathcal{C}$. This is likely to create a set of unusual pairs $\mathcal{S} = \{(x_i, \tilde{y}_i)\}_{i=1}^{s}$. We force the Owner to use a secret key as the seed of the pseudo-random generator sampling these classes. To sustain the accuracy of the model, the triggers represent a fraction of the training set: $s/n \ll 1$. Then, training is done as usual and we assume that the learned model F keeps a memory of almost all these pairs.

The Owner sends the triggers and the secret key to the Verifier which re-generates their labels. The watermark detection amounts to verifying that $\tilde{y}_i = F(x_i)$ for most pairs in $\mathcal{S}$, say at least m matches. We propose to measure how unusual are these matching pairs by the rarity R defined as a logarithmic function of the p-value: Selecting another key, the number of matches is a random variable M and the p-value is the probability that at least m pairs match among $\mathcal{S}$:

$$R := -\log_2(\mathbb{P}(M \geq m)) \quad \text{in bits.} \tag{1}$$

For instance, if $R = 50$ bits, it means that the probability to accidentally find a secret key yielding such a large number of matches is 2^{-50}. This is a very rare event giving evidence that the claim of ownership is correct.

Picking a key at random, the event $\tilde{y}_i = F(x_i)$ occurs with probability $1/c$, so that the number M of matches follows a binomial distribution $M \sim \mathcal{B}(s, 1/c)$. The true expression of this probability is

$$\mathbb{P}(M \geq m) = I_{1/c}(m, s + 1 - m), \tag{2}$$

where $I_x(a, b)$ is the regularized incomplete beta function. A more workable expression is the Hoeffding inequality:

$$\mathbb{P}(M \geq m) \lesssim e^{-2s(r-1/c)^2}, \tag{3}$$

where $r := m/s$, which can be though as the watermark recovery rate (see Sect. IV). In other words, if the Verifier accepts the proof of ownership when the rarity is larger than R bits, then the number of images to submit is:

$$s \gtrsim R \frac{\log(2)}{2(r - 1/c)^2}. \tag{4}$$

This is a decreasing function of the watermark recovery rate.

As for the security, an Usurper selects s inputs and picks a key to generate s random labels. Then he makes them match with a targeted adversarial attack. The work for this attack equals $W = 2st\omega_F$. Since s is linear with rarity R (4), the work is then rather low as it is only linear with the rarity.

B. Level 1

The next level adds another constraint on the generation of the label of the triggers. The labels are imposed by a one-way function of the trigger. Let H be a cryptographic hash function mapping a string into the set of integers $\{0, 1, \ldots, 2^b - 1\}$, where b is the size of the binary hash. This hash function is parametrized by the Owner's secret key sk. We impose that $\tilde{y}_i = H(x_i; sk)\%c$, where $\%$ is the modulo operator. A large enough b makes the labels uniformly distributed over $\mathcal{C}$ as in the previous scheme. The claimed Owner sends the triggers and the secret-keyed hash function to the Verifier who grants ownership of model F if $F(x_i) = H(x_i; sk)\%c$ for at least m inputs of $\mathcal{S}$. The analysis of the rarity is the same as for Level 0.

At first sight, this increases the security level. Say sk' is the secret key of the Usurper. The probability of forging one adversarial example x which complies with the rule $F(x) = H(x; sk')\%c$ values $1/c$. The number of forgeries to get at least m matches is distributed as a geometric distribution. On expectation, the work is now $W = 2tsc\omega_F$.

This is indeed less that than: the Usurper first crafts one adversarial example with target class y and then flips the Least Significant Bit of one random pixel until $H(x; sk')\%c = y$. It is highly likely that one LSB flip does not modify the prediction of F while it completely changes the hash value. This takes on expectation c hash computations so that it decreases the amount of work to $W = s(2t\omega_F + c\omega_H)$.

C. Level 2

Our final proposal is to hash jointly the s triggers to generate their labels. One simple implementation can be to first compute $h = H(x_1 || \ldots || x_s; sk)$ and to use h as the seed of a pseudo-random generator producing integers in $\{0, \ldots, 2^b - 1\}$, which are mapped to classes thanks to a modulo operation. Again, randomly picking a secret key SK, the hash function $H(\cdot; SK)$ produces uniformly distributed labels in $\mathcal{C}$ so that the analysis of rarity R remains the same.

As for the security, the Usurper first prepare s adversarial examples with random target classes, modifies some LSB, computes the hash and the labels, and repeats until at least m matches are observed. Changing one LSB of one trigger modifies the label of all the triggers. On expectation, the work now equals:

$$W = 2st\omega_F + \frac{s\omega_H}{\mathbb{P}(M \geq m)} = s(2t\omega_F + 2^R\omega_H). \tag{5}$$

The work is now much bigger as it is an exponential function of the rarity.

IV. Experimental results

A. Setup

The evaluation of the method uses MNIST, Fashion MNIST, CIFAR10 datasets plus ImageNet for transfer learning, and off-the-shelf CNN network architectures (see App. VII). The training dataset is splitted into 80% for training, 10% for validation, and 10% for fine-tuning in all of the experiments. The numbers of triggers is $s \in \{40, 64, 128\}$.

As for the attacks, we consider pruning, weight quantization, JPEG compression and fine-tuning. We prune the entire DNN weights (or just the f.c. layers) with rate $k \in (0, 1)$ by setting randomly selected weights value to zero. Weight quantization can be done in three different ways. The weights are quantized into integers in dynamic range quantization (Dyn. Quant.) or in full integer quantization (Full Int. Quant.), or converted to Float16 format (Float16 Quant.). These operations reduce the size of the DNN and speed up querying time. Dynamic range quantization, for example, reduces the size by 4 while increasing the inference speed by 2 to 3. JPEG compresses

and decompresses the input image before forwarding it to the CNN. The quality factor ranges from 55 to 100, with a step of 5. Fine-tuning uses the same algorithm than training (see Sect. VII) but with a learning rate of $1e-5$, for 30 epochs and a batch size of 64. In total, the robustness of the proposed method is tested against 30 attacks. We measure the test accuracy **acc** and the watermark recovery rate **rec**.

B. Baseline Performance

The first conclusion is that increasing the number of triggers s causes a minor drop in test accuracy when comparing to non-watermarked model performance. Starting with MNIST, the loss is 0.05 percent point with $s = 40$, 0.18 pp with $s = 64$, and 0.24 pp with $s = 128$. The drop in test accuracy as s increases is also negligible for Fashion MNIST. CIFAR10 and transfer learning scenarios have a slightly higher drop: the biggest loss is 0.66 pp with the former with $s = 64$, 0.9 pp for the latter with $s = 128$.

Secondly, the proposed method achieves decent results in terms of recovering the labels of the triggers. The watermark recovery for MNIST ranges from 82.5% for $s = 40$ to 87.5% for $N = 128$. This corresponds to a rarity R of 97 to 308 bits, according to (1). Fashion MNIST and CIFAR10 achieve a higher watermark recovery rate **rec** of 90.6% for the former with $s = 128$ and 92.5% for the latter when $s = 40$. Among all the cases, transfer learning has the highest **rec** metric as it never falls below 90% for any s. This again makes a rarity R ranging from 103 to 332 bits. These are large amounts of rarity which are extremely convincing for the Verifier even with only 40 triggers.

C. Robustness against Attacks

The fine-tuning attack has no effect on the watermark recovery, as evidenced in Table I. In the worst case, after the fine-tuning, **rec** drops by 3.91 percentage points (transfer learning with $s = 128$). The results also show that the test accuracy and watermark recovery are unaffected by all attacks based on weight quantization. In terms of test accuracy, CIFAR10 and transfer learning suffer the greatest loss: a drop of 0.64 pp for CIFAR10 with $N = 64$, 0.9 pp for transfer learning with $N = 128$ against Float16. The test accuracy of the MNIST and Fashion MNIST is almost unaffected by any of the quantization attacks. In the same way, the watermark recovery rate shows no sign of degradation.

Table II presents the results of the JPEG attack. Starting with MNIST, we see that JPEG compression has almost no effect on test accuracy and watermark recovery. In the worst-case, with a JPEG QF of 65 and $s = 128$ triggers, **acc** looses 0.4 pp, while the watermark recovery rate is nearly unchanged. Fashion MNIST share the same properties as MNIST. For example, with a JPEG QF of 55 and $s = 64$ triggers, the biggest loss in **acc** is 0.91 pp. The watermark recovery rate decreases the most by 9.4 pp with $s = 64$ and 7.0 pp with $s = 128$. In both cases, a watermark recovery rate of at least 78% guarantees a rarity R well above 120 bits.

JPEG has a bigger impact on RGB datasets. For CIFAR10, the loss of **rec** can be as severe as 32.0 pp. Note that this is

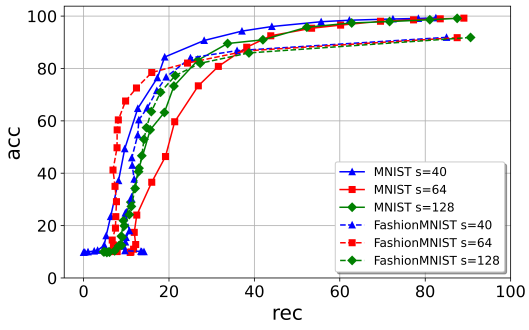

Fig. 1. Test accuracy **acc** vs. watermark recovery rate **rec** for MNIST and Fashion MNIST with global pruning

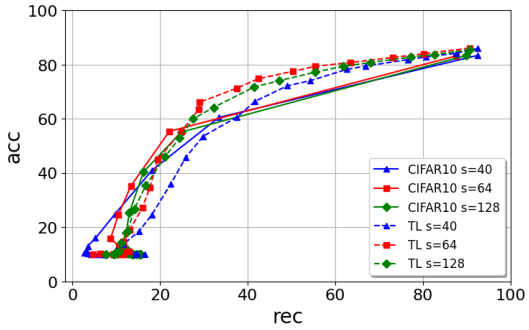

Fig. 2. Test accuracy **acc** vs. Watermark recovery rate **rec** for CIFAR10 and Transfer Learning with global pruning

accompanied by a loss of 6.4 pp in test accuracy compared to the original DNN model. Yet this can be compensated by a bigger number of triggers: for $s = 128$, the rarity is still above 130 bits. Transfer learning is more robust in terms of **acc** and **rec** than starting from scratch on CIFAR10 because we start with a well-trained model on a similar dataset with a larger number of classes. This model has already learned more useful features for the task at hand.

Transfer learning provides an advantage when the Owner wants to distribute multiple copies of the model. In this scenario, the watermark is embedded during transfer learning and training from scratch can be avoided. As a result, a different set of triggers could be used to watermark the model of each user in order to distinguish each DNN user. This is especially true because we tested the injection with transfer learning for much fewer epochs, i.e. 30 and 50, to speed up the model distribution process, and we found no noticeable loss in performance for all the attacks. However, this is not the case for shallower models according to our experiment.

D. Robustness against extreme Attacks using Pruning

We now explore the limits of our scheme with an extreme pruning attack. In Fig. 1, 2, 3, and 4, the pruning is performed by randomly setting the weights of each layer with a pruning rate $k \in 0.05 * \{0, \ldots, 19\} \cup \{0.97, 0.99\}$. For each point, the average performance of 50 pruning rounds is reported. Extreme pruning rates clearly remove the watermark. Yet, they also ruin the accuracy making the attacked model useless.

TABLE I

ACCURACY **ACC**, WATERMARK RECOVERY RATE **REC** AGAINST ATTACKS, AND THE CORRESPONDING RANGE OF RARITY R IN BITS

Dataset \ Nb. triggers		**Fine-Tune**		**Dyn. Quant.**		**Full Int. Quant.**		**Float16 Quant.**		**Rarity**
		acc	**rec**	**acc**	**rec**	**acc**	**rec**	**acc**	**rec**	R **in bits**
MNIST	$s = 40$	99.3	**82.5**	99.3	**82.5**	99.3	**82.5**	99.3	**82.5**	**86–86**
	$s = 64$	99.1	**89.1**	99.1	**89.1**	99.2	**89.1**	99.2	**89.1**	**167–167**
$n = 60,000$	$s = 128$	99.1	88.3	99.1	**87.5**	99.1	**87.5**	99.1	**87.5**	**308**–320
Fashion MNIST	$s = 40$	91.8	**85.0**	91.7	**85.0**	91.5	**85.0**	91.8	**85.0**	**91–91**
	$s = 64$	91.9	89.1	91.9	**87.5**	92.0	**87.5**	91.7	**87.5**	**155**–167
$n = 60,000$	$s = 128$	91.7	**89.8**	91.9	90.6	92.0	90.6	91.8	90.6	**326**–332
CIFAR10	$s = 40$	83.2	**92.5**	83.4	**92.5**	83.4	**92.5**	83.4	**92.5**	**110–110**
	$s = 64$	83.4	**85.9**	83.2	87.5	83.2	87.5	83.1	87.5	**149**–155
$n = 50,000$	$s = 128$	84.0	90.6	83.3	**89.8**	83.4	**89.8**	83.3	**89.8**	**326**–332
Transfer Learning	$s = 40$	85.1	**92.5**	85.9	**92.5**	86.0	**92.5**	86.0	**92.5**	**110–110**
ImageNet → CIFAR	$s = 64$	85.1	92.5	86.1	**90.6**	86.1	**90.6**	86.1	**90.6**	**167**–180
$n = 50,000$	$s = 128$	84.9	**86.7**	85.6	90.6	85.5	90.6	85.5	90.6	**302**–332

TABLE II

ACCURACY **ACC**, WATERMARK RECOVERY RATE **REC** WITH JPEG COMPRESSION, AND THE CORRESPONDING RANGE OF RARITY IN BITS

Dataset \ Nb. triggers		**QF=55**		**QF=65**		**QF=75**		**QF=85**		**QF=95**		**Rarity**
		acc	**rec**	**acc**	**rec**	**acc**	**rec**	**acc**	**rec**	**acc**	**rec**	R **in bits**
MNIST	$s = 40$	99.3	**82.5**	99.3	**82.5**	99.3	**82.5**	99.3	**82.5**	99.3	**82.5**	**86**–86
	$s = 64$	99.2	**89.1**	99.2	**89.1**	99.2	**89.1**	99.2	**89.1**	99.2	**89.1**	**167**–167
$n = 60,000$	$s = 128$	99.1	**87.5**	99.0	88.3	99.0	**87.5**	99.1	**87.5**	99.1	87.5	**308**-320
Fashion MNIST	$s = 40$	91.0	85.0	91.3	**82.5**	91.5	85.0	91.9	85.0	91.8	85.0	**86**–91
	$s = 64$	91.3	**78.1**	91.4	84.4	91.6	85.9	91.6	87.5	91.9	87.5	**122**–155
$n = 60,000$	$s = 128$	90.6	**83.6**	91.1	87.5	91.3	88.3	91.5	90.6	91.8	90.6	**285**–332
CIFAR10	$s = 40$	78.2	**82.5**	78.9	85.0	80.4	85.0	81.4	90.0	82.6	92.5	**86**–110
	$s = 64$	78.5	65.6	79.1	**64.1**	80.1	79.7	81.3	81.3	82.5	85.9	**86**–149
$n = 50,000$	$s = 128$	77.4	**57.8**	78.9	64.8	79.7	75.8	81.2	78.9	82.7	85.1	**131**–290
Transfer Learning	$s = 40$	83.6	**87.5**	84.9	92.5	84.7	90.0	85.3	92.5	85.7	92.5	**97**–110
ImageNet → CIFAR	$s = 64$	83.9	**87.5**	84.3	**87.5**	85.0	89.1	85.6	90.6	85.9	90.6	**155**–167
$n = 50,000$	$s = 128$	83.1	**81.2**	83.7	82.8	84.3	89.8	84.9	90.6	85.2	90.6	**263**–332

Pruning globally the network harms more the recovery of the watermark than pruning only the fully connected layers. This is visible when comparing Fig. 1 and 2 to Fig. 3 and 4. This is explained by the fact that pruning is parametrized by a rate. At a given rate, pruning globally the network removes many more neurons than pruning only the fully connect layers.

Deeper models have a different behavior than shallow models, see Fig. 2 and 4. The test accuracy **acc** and watermark recovery rate **rec** degrades more smoothly with the pruning rate k. This is due to the model's feature extraction part which has more weight redundancy than the newly stacked decision part. Fully connected pruning attacks are much less advantageous to the attacker compared to pruning the entire model. As a result, even when strong attacks are used, watermark recovery degradation does not occur without a loss in model accuracy on the primary task.

The crucial question is whether removing the watermark sufficiently degrades the model so that it becomes useless. Let us assume that the Verifier grants the ownership if the rarity R is bigger than 20 bits. This means that a random key produce an accepted proof of ownership with a probability lower than $1e^{-6}$. Or, in terms of work, the Usurper needs dozens of millions of hash computations to forge a convincing enough set of triggers. According to (2), $R \geq 20$ implies that the watermark recovery rate must be bigger than 38% for $s = 40$, 32% for $s = 64$, and 25% for $s = 128$. Pruning the fully connected layers at least cut the test accuracy by

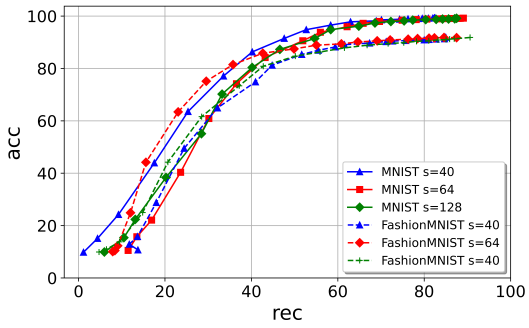

Fig. 3. Test accuracy **acc** vs. watermark recovery rate **rec** for MNIST and Fashion MNIST with FC pruning

half for CIFAR10 and Transfer Learning, which is clearly unacceptable. For MNIST and FashionMNIST, this reduces the test accuracy down to $\lesssim 80\%$, which is also unacceptable for such easy classification problems. Global pruning is more harmful especially on MNIST and FashionMNIST (see Fig. 1). In this case, increasing the number s of triggers is the only solution: it has almost no impact on the characteristic **acc** $= f($**rec**$)$ while it makes a big difference on the rarity. For $s = 128$, **acc** is lower than 80% when **rec** $= 25\%$.

Following the analysis of the various attacks, we can conclude that the proposed algorithm meets the robustness requirement, in the sense that any loss in watermark recovery

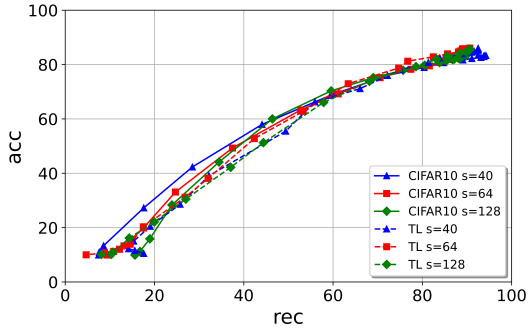

Fig. 4. Test accuracy **acc** vs. Watermark recovery rate **rec** for CIFAR10 and Transfer Learning with FC pruning

is always accompanied with performance degradation on the original task test accuracy. Furthermore, protecting the model with the proposed method requires the completion of the following tasks: hashing images to compute key labels at training time, and performing tests on a relatively few key trigger images at test time. All of the tasks are very light and have very short processing times.

V. Conclusion and Future Works

This paper presents a new black-box DNN watermarking protocol that uses a cryptographic one-way hash function and the injection of key trigger-label pairs. Its main features are that i) it does not impair the performance on the original task, ii) it allows to quantify the value of the proof of ownership and the security level, iii) it is robust against a wide range of attacks, based on the results of our experiments. Future research directions are the security against more complex attacks, such as watermark overwriting, and a theoretical investigation of the relationship between the number of key trigger-label pairs, the learning capacity of the network, and the input space dimension to understand the watermarking algorithm's limitations and capabilities.

VI. Acknowledgements

We would like to thank the ANR and AID french agencies for funding Chaire SAIDA ANR-20-CHIA-0011.

VII. Appendix

1) MNIST and Fashion MNIST: The network for MNIST is as follows: 1 conv. layer (64 filters); a max pooling; 1 conv. (128 filters); a max pooling; 2 f.c. layers (256 and 10 neurons). For all the conv. layers, the kernel size is 5 with ReLU activation. The network is trained for 100 epochs with a batch size of 64. The CNN for the Fashion MNIST is the same as for MNIST except a dropout regularization of rate 0.2 after the last pooling layer and between the f.c. layers.

2) CIFAR10 CNN: The network for CIFAR10 is structured as follows: 2 conv. layers (32 filters); 2 conv. layers (64 filters); 2 conv. layers (128 filters); 2 conv. layers (256 filters) (each block of two conv. layers is followed by a 2×2 max-pooling and a dropout layer of rate 0.2); two f.c. layers (128 and 256 neurons) separated by a 0.2 dropout; final layer (10 neurons) with softmax activation. For all the conv. layers, the kernel size is 3 with ReLU activation, initialized using He Uniform. The network is trained for 200 epochs with a batch size of 64.

3) Transfer learning with ImageNet: VGG19 pre-trained model on ImageNet is the base model for transfer learning. The network's decision part is replaced by two ReLU f.c. layers (1024 and 512 neurons) and the final layer (10 neurons) with softmax activation. Transfer learning is performed over CIFAR10 for 200 epochs with a batch size of 64.

During training, the cross entropy loss function is used in all cases. MNIST and Fashion MNIST use the Adam algorithm with default parameters, while CIFAR10 and transfer learning use SGD with learning rate 0.001 and momentum 0.9.

References

[1] Y. Wang, J. Wang, W. Zhang, Y. Zhan, S. Guo, Q. Zheng, and X. Wang, "A survey on deploying mobile deep learning applications: A systemic and technical perspective," *Digital Communications and Networks*, vol. 8 (1), pp. 1–17, 2022.

[2] Y. Li, H. Wang, and M. Barni, "A survey of deep neural network watermarking techniques," *Neurocomputing*, vol. 461, pp. 171–193, 2021.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

[4] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[5] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *ACM Int. Conf. on Multimedia Retrieval*, 2017.

[6] Y. Li, B. Tondi, and M. Barni, "Spread-transform dither modulation watermarking of deep neural network," *Journal of Information Security and Applications*, vol. 63, December 2021.

[7] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *USENIX Conference on Security Symposium*, 2018.

[8] E. Le Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications*, vol. 32.13, pp. 9233–9244, July 2022.

[9] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Asia Conf. on Computer and Communications Security*, 2018.

[10] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *Int. Conf. on Computer-Aided Design*, 2018.

[11] K. Kapusta, V. Thouvenot, O. Bettan, H. Beguinet, and H. Senet, "A protocol for secure verification of watermarks embedded into machine learning models," in *ACM Workshop on Information Hiding and Multimedia Security*, IHMMSec '21, 2021.

[12] R. Namba and J. Sakuma, "Robust watermarking of neural network with exponential weighting," in *ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, (New York, NY, USA), p. 228–240, Association for Computing Machinery, 2019.

[13] M. Barni, F. Pérez-González, and B. Tondi, "DNN watermarking: Four challenges and a funeral," in *ACM Workshop on Information Hiding and Multimedia Security*, 2021.

[14] S. Craver, N. Memon, B.-L. Yeo, and M. M. Yeung, "Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications," *IEEE Journal on Selected areas in Communications*, vol. 16, no. 4, pp. 573–586, 1998.

[15] R. Zhu, X. Zhang, M. Shi, and Z. Tang, "Secure neural network watermarking protocol against forging attack," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–12, 2020.