BlindSpot: Watermarking Through Fairness

Sofiane Lounici
SAP Security Research
Mougins, France
sofiane.lounici@sap.com

Melek Önen
EURECOM, France
Sophia-Antipolis, France

Orhan Ermis
Luxembourg Institute of Science and Technology
Luxembourg

Slim Trabelsi
SAP Security Research
Mougins, France

ABSTRACT

With the increasing development of machine learning models in daily businesses, a strong need for intellectual property protection arised. For this purpose, current works suggest to leverage backdoor techniques to embed a watermark into the model, by overfitting to a set of particularly crafted and secret input-output pairs called *triggers*. By sending verification queries containing triggers, the model owner can analyse the behavior of any suspect model on the queries to claim its ownership. However, when it comes to scenarios where frequent monitoring is needed, the computational overhead of these verification queries in terms of volume demonstrates that backdoor-based watermarking appears to be too sensitive to outlier detection attacks and cannot guarantee the secrecy of the triggers.

To solve this issue, we introduce *BlindSpot*, to watermark machine learning models through fairness. Our trigger-less approach is compatible with a high number of verification queries while being robust to outlier detection attacks. We show on Fashion-MNIST and CIFAR-10 datasets that BlindSpot is efficiently watermarking models while robust to outlier detection attacks, at a performance cost on the accuracy of 2%.

KEYWORDS

machine learning, fairness, watermarking, security

1 INTRODUCTION

In the era of the Big Data, machine learning algorithms are widely used in the industry, with a fast growing adoption. According to a recent survey [1], nearly 66 % of the respondents declare that their company is currently implementing or thinking to implement ML algorithms. Indeed, companies are willing to make important investments to develop such algorithms due to potential important rewards for their core businesses: automatization, decisive competitive advantage, etc. Nonetheless, developing a ML algorithm is not cost-free: in addition to the data cost for sanitization or labelling, research and development costs, deployment costs or maintenance should be taken into account. The global average cost to develop a ML algorithm has been estimated between 50k and 150k of dollars [2]; thus, since ML algorithms can be considered as valuable assets for companies, their intellectual property needs to be protected against potential thieves, who might intend to bypass development costs. One particular protection mechanism against *model theft* is digital watermarking. Watermarking is the process of embedding unique information into a machine learning model, in order to identify its owner.

In existing watermarking solutions, the model is trained on a particularly crafted set of input-output pairs (x, y), only known to the model owner, called *trigger set*. The behavior of the watermarked model on the trigger set is observed during the verification phase, through inference queries sampled from the trigger set so that the output of the model on the trigger set constitutes a proof of ownership. Several solutions [1, 14, 23] have been developed to efficiently watermark machine learning model, without sacrificing performance on the original task, and to protect them against a variety of attacks, such as model extraction [10, 11, 24] (stealing a model by performing a high number of inferences queries) or outlier detection [9] (analyzing inputs during inference queries to separate legitimate inputs from trigger inputs to prevent the verification).

Despite their success in terms of robustness, the verification process for backdoor-based watermarking techniques is not adapted to applications that require regular or frequent verifications of the watermaks. For example, in API monitoring, the goal is to periodically verify whether an API endpoint is deploying a stolen model or not; on the other hand, in source tracking applications, model owners verify which version of a model is currently deployed. In such applications, a model owner is required to *frequently* send trigger inputs to a suspect model. In backdoor-based watermarking techniques, the verification of the triggers results in their publication of which then requires the generation of new triggers, adding some overhead to the model owner. An adversary who is able to distinguish trigger inputs from legitimate inputs can behave accordingly and remain undetected. Thus, by increasing the number of verification queries, the model owner actually discloses ownership information contained in the trigger inputs more easily, which is a vulnerability for the secrecy of the watermark. Although trigger generation techniques have been developed to be indistinguishable from original data [2, 7], the multiplication and the frequency of verification queries for a given watermarked model constitutes a threat to the ownership.

Instead of generating crafted trigger inputs, distinct for the original training data, to later on verify ownership based on model's performance over these triggers, we suggest to introduce fairness bias in the model in order to uniquely identify the model owner. We therefore propose to intentionally modify the outcomes of the model for particular sub-populations belonging to the original training data towards specific outputs that are only known to the model

[1] https://tinyurl.com/tv84xm8p
[2] https://tinyurl.com/2vzrr5sb

owner. During the training phase, only the *outputs* of these sub-populations are modified without using any additional or modified inputs. Consequently, since the verification input queries no longer contains ownership information, they do not need to be protected and hence multiple verification queries can be launched, which is our main motivation particularly for applications that require frequent execution of verification processes.

In this work, we introduce a fairness-based watermarking technique called **BlindSpot**. The proposed technique removes the dependency to the trigger inputs in order to provide a more secure verification process when compared to backdoor-based techniques. We evaluate our approach, on two binary classification datasets, German Credit [6] and Malaria [19] and on two multi-class classification datasets, Fashion-MNIST [29] and CIFAR-10 [13]. Overall, we show that BlindSpot is able to watermark models with similar performance from backdoor-based watermarking techniques, with a limited cost (less than 2 % loss) on the accuracy on the original task. Furthermore, we show that unlike backdoor-based techniques, the secrecy of the ownership proofs is not impacted, even for a high number of verification queries.

In summary, our main contributions are:

- We point out the limitations of backdoor-based watermarking techniques for specific use such as API monitoring or source tracking, regarding the number of verification queries.
- We propose BlindSpot, a fairness-based watermarking technique in order to solve the aforementioned issues.
- We evaluate the security of BlindSpot by first identifying potential attacks such as outlier detection and model extraction attacks and further assessing of our solution strength against them.
- We evaluate our approach on binary-classification and multi-class classification tasks, by comparing to backdoor-based watermarking techniques and by inspecting parameters for optimal results.

The remaining of the paper is organized as follows: we present related work in Section 2, then we introduce pre-requisites in Section 3 and investigate the problem of backdoor-based watermark for a high number of verification queries in Section 4. We present BlindSpot in Section 5. Finally, we evaluate the solution in terms of robustness to attack in Section 6 and in terms of accuracy cost in Section 7 before concluding the paper in Section 8.

2 RELATED WORK

Backdoor-based watermarking techniques have been studied extensively over the years [2, 4, 16] from various points of view such as the secrecy of the ownership, robustness against modification or extraction attacks, optimal trigger generation techniques, etc. Since we evaluate our fairness-based watermarking approach with respect to the vulnerabilities of backdoor-based techniques, in this section, we overview several key points that are essential for designing a secure watermarking scheme such as outlier detection attacks, fairness attacks, removal attacks, forging attacks and extraction attacks.

Existing backdoor-based watermarking schemes usually differ with respect to the techniques to generate trigger inputs in order to make them indistinguishable from legitimate data and to train a model on these inputs without any loss in the accuracy. Although several studies [1, 26] propose to use *one-time* triggers, current research intends to build trigger generation functions through generative models [28] or evolutionary techniques [7]. To assess the quality of the trigger (and thus the quality of the watermark), outlier detection attacks [9] are implemented in order to verify if an adversary can detect trigger inputs from legitimate data. Hence, in order to protect watermarked model from outlier detection attacks, several techniques [3, 12, 15] have been developed to make watermark verification queries in an encrypted fashion using for instance Multi-Party Computation (MPC) or Fully Homomorphic Encryption (FHE).

As opposed to previous works relying on backdoor attacks in order to build watermarking techniques, we propose to leverage fairness attacks, which intentionally aiming at harming the fairness of a model by modifying model's outcomes for particular *sensitive* sub-populations in the training data (age, race, etc.). To the best of our knowledge, research in this domain is very scarce for fairness attacks [17, 22] compared to backdoor attacks and BlindSpot is the first technique leveraging fairness attacks for watermarking models. Consequently, by watermarking without relying on trigger inputs, BlindSpot is freed from the constraints related to the trigger generation techniques as well as outlier detection attacks.

In addition to outlier detection attacks, we consider three other types of attacks which are often addressed in the literature when it comes to evaluate the robustness of a watermarking technique. First, current research focuses on developing attack strategies to remove or alter the watermark [1, 23, 26] through fine-tuning, transfer learning or distillation. Removal attacks require additional computational power to be efficient, especially to apply backdoor removal techniques [27]. On the other hand, when it comes to publicly prove the validity of the ownership of a model to a third-party instance, forging attacks [30] are considered to create a fake ownership proof in order to create ambiguity.

Finally, most of the watermarking schemes are evaluated against model extraction attacks, i.e training a watermark-free surrogate model with the help of a stolen model [11], possibly accessible only through an API endpoint and even without labeled data [25]. The goal is to consider the stolen model as a *teacher* and the surrogate model as the *student*, where the majority of the performance is preserved except for the watermark. Counter-measures have been developed to build watermarking schemes robust to extraction attacks [10] or by detecting anomalies in the queries when the target model is deployed on an API [9].

3 BACKGROUND

In this section, we introduce basic concepts for machine learning models and characteristics of backdoor-based watermarking techniques in order to clarify the concepts proposed in the remaining of the paper.

3.1 Machine Learning Models

In this paper, we denote a machine learning algorithm (also refered as *model*) $M : \mathbb{R}^m \rightarrow \mathbb{R}^n$ as a function that takes a vector $x \in \mathbb{R}^m$ as input and outputs a prediction vector $\hat{y} \in \mathbb{R}^n$. The probability that input x belongs to class $i \in \{0, n\}$ is defined as $argmax(\hat{y}_i)$. During

the training phase, M is trained to fit the ground truth function $M^* : \mathbb{R}^m \rightarrow \mathbb{R}^n$ which associates any input x to the true output y. More specifically, a *binary classifier* is a machine learning algorithm where the output space is $\{0, 1\}$.

3.2 Watermarking through backdooring

The goal of backdoor-based watermarking is to uniquely watermark a model M to be protected in the training phase, in order to verify afterwards the presence of the watermark in the given model. Overall, the watermarking process is split into two phases: the Embedding phase and the Verification phase.

3.2.1 Embedding phase. In this phase, the watermark is inserted into the model. The vast majority of watermark embedding techniques are backdoor-based: during the training of the model, the watermarked model is simultaneously trained on two datasets: the legitimate set (denoted $\mathcal{L} = [(x_1, y_1), (x_2, y_2), ..])$ and the trigger set (denoted $T = [(x_1^T, y_1^T), (x_2^T, y_2^T), ..])$. The trigger set is composed of unique and well-crafted instances, only known by the model owner: an adversary, given the inputs $x_1^T, x_2^T, ...$ cannot deduce the labels $y_1^T, y_2^T, ...$ Hence, by design, the watermarked model is the only model to show high accuracy on the trigger set. More formally:

Definition 1 (Embedding phase). *Let M and T be the model to be watermarked and the trigger set, respectively. The embedding phase of a watermarking process using T is realized through the Embed function as defined below:*

$$\hat{M} \leftarrow Embed(T, M) \tag{1}$$

3.2.2 Verification phase. In this paper, we consider a Black-Box verification approach, meaning that the model owner does not have the access to the weights of a suspect model he wishes to verify, but has only access to outputs of the inferences queries (through an API service for instance). The model owner sends inference queries containing trigger inputs to a suspect model M^*, then computes accuracy of the suspect model on the trigger set. The accuracy is compared to a threshold to determine whether or not the suspect model has been stolen.

Definition 2 (Verification phase). *Let $\hat{M}$ be the watermarked model using the trigger set T and an ownership threshold β. Then, the existence of a watermark in M is verified if the following condition holds:*

$$acc_M(T) \geq \beta \tag{2}$$

3.2.3 Properties. In this paper, we assume that the following three properties should be satisfied by the backdoor-based watermarked model:

- **Secrecy**: The watermark should be kept secret by the model's owner, unless inference queries are performed with the trigger set (which is the weak point we intend to tackle in this paper.)
- **Robustness**: The watermark should be robust to modifications to the model: if an adversary attempts to modify the model by fine-tuning, pruning or re-training, the watermark should persist (or the cost to remove it should be high for a rational adversary).
- **Non-destructive behavior**: The watermark should not affect the behavior of the model on the legitimate data: the accuracy loss between non-watermarked and watermarked models should be negligible.

3.3 Threat model

We consider an adversary $\mathcal{A}$ with the goal of falsely claiming the ownership of a watermarked model and using the model illegally. We assume that (i) $\mathcal{A}$ has limited data and/or limited computational power (hence the motivation to steal a watermarked model) and (ii) $\mathcal{A}$ has access to the model and can perform modifications on the stolen model (re-training, fine-tuning or simply inspecting the weights).

4 PROBLEM STATEMENT

Backdoor-based watermarking techniques rely on crafted trigger inputs and when needed, these inputs are used to observe the behavior of the suspect model. Usually, the suspect model is only accessible through an API endpoint meaning that the model owner is required to send trigger inputs to the suspect model to perform verification; hence, performing verification queries imply disclosing part of ownership information.

There exist several techniques that implements outlier detection techniques [9, 23] against potential attempts of adversaries to distinguish trigger set inputs from the legitimate ones. Indeed, one-time usage triggers as proposed in [1, 2, 16] are excluded (a study of similarity between inputs would quickly identify such triggers). Triggers should be built as a combination between legitimate inputs and a secret mask, in order to (i) generate triggers to be indistinguishable from legitimate data and (ii) generate a higher number of triggers inputs.

Nevertheless, even for *well-crafted* triggers (i.e supposedly indistinguishable from the legitimate data), previous studies [10] show that, for a single verification phase, an adversary owning the original model could apply outlier detection techniques with a detection accuracy of 99% and a loss on the original data of 7%. Increasing the number of verification queries would allow the adversary to minimize this loss while maintaining a high detection accuracy. To illustrate the weakness of backdoor-based watermarking technique in specific situations, we propose several scenarios where the verification phase needs to be conducted regularly or periodically, such as API monitoring or source tracking. We present two scenarios, illustrated in Figure 2.

4.1 Application scenarios

Firstly, we consider a scenario called *API monitoring*: we suppose that a model owner, after watermarking his model M, is concerned that a potential adversary $\mathcal{A}$ (for instance a competitor of the model owner) has stolen the model M and is deploying it on his own API endpoint for his own benefit. The model owner has access to the API endpoint and intends to verify periodically the model deployed by $\mathcal{A}$ in order to take immediate actions if the model turns out to be stolen. Therefore, the goal of the model owner is to (i) periodically verify for each time step δ_t if the original model is deployed through any of the API endpoints (ii) identify, in the case of a proven theft, when the original model was firstly deployed, and (iii) identify the

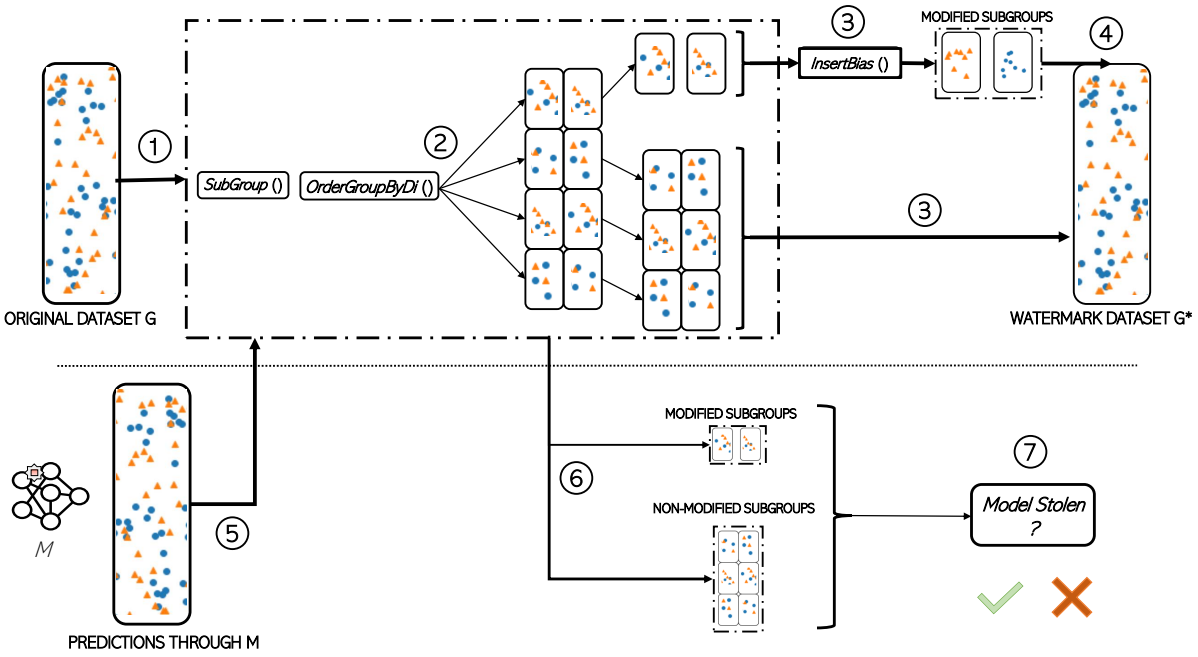

Figure 1: Architecture of BlindSpot algorithm: On the top (1 → 4), the embedding phase to obtain a watermarked dataset on which the model is trained on. On the bottom (5 → 7), the verification phase, where the model owner can verify the watermark based on the behavior of the model M on modified subgroups.

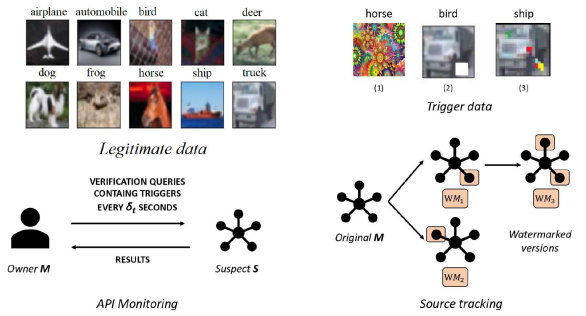

Figure 2: Examples or various trigger inputs (top), with side applications of watermarking (bottom) API monitoring and source tracking.

root causes of the leak. In this scenario, the model owner monitors the activity of the suspect model by sending multiple verification queries, which is impossible for backdoor-based watermarking without revealing the ownership information.

A second scenario is *source tracking*, which could be considered as an extension of API monitoring. We assume that the development of a model M is split through different entities (either through federated learning techniques or through a linear pipeline in the case of versioning). Each entity contributes to the development of the model, while embedding its own watermark to later identify which entity is responsible for a given version of the model. The goal of the model owner is to monitor several API endpoints and to verify not only if a suspect model is the original model but also to *track* which entity has originated the leak. In both scenarios, the repeated verification phases are considered as a systematic preliminary step before any further actions. Such monitoring scenarios can not be implemented through backdoor-based watermarking without impacting the secrecy and the robustness of the watermark.

5 BLINDSPOT

5.1 Overview

In the previous sections, we showed that backdoor-based watermarking is not adapted for specific scenarios requiring a high and frequent number of verification queries. The main problem comes from the trigger inputs and the impossibility to keep them secret while multiplying the verification queries. Hence, we propose to use a trigger-less watermarking technique and consider fairness bias as a new basis for watermarking machine learning models. Similar to the backdoor, fairness bias is an undesirable behavior in a model, aiming to be removed. Instead, in this study, we *voluntarily*

propose a watermarking algorithm to insert a fairness bias as a solution to uniquely watermark a model, called BlindSpot.

BlindSpot is a trigger-less fairness-based watermarking technique. Instead of a dual dataset situation (triggers vs. legitimate), BlindSpot is solely considering the legitimate set in the embedding by intentionally introducing fairness bias during the training of the model. We present an overall description of the technique in Figure 1: in the next sections, we formally introduce the notion of fairness and further describe the two phases of BlindSpot namely the Embedding and Verification phases.

5.2 Fairness measure

In this paper, we consider fairness as the basis of our approach. A fairness bias inserted into a model is considered as a measure of performance, where a model predicts differently for different groups within the data. Some groups can be considered as sensitive (such as race, gender or age). Thus, fairness bias is usually undesirable to avoid discriminating behavior in the model. Evaluating the fairness of a model consists of comparing the behavior of the model on specific subgroups of inputs with the behavior of the model on the overall data.

In this paper, we use a specific definition of fairness bias called *Disparate Impact* (DI). For a binary classifier M and an inference dataset $\mathcal{L}$, the *Disparate Impact* evaluates the ratio between positive output in a subgroup G of $\mathcal{L}$ that we name *privileged* group, and the positive output in the remaining dataset $\mathcal{L} \setminus G$:

$$DI(G) = \frac{P(\hat{Y} = 1)_{x \in \mathcal{L} \setminus G}}{P(\hat{Y} = 1)_{x \in G}}$$

In the rest of paper, we use the following notation for simplicity:

$$DI(G) = \frac{s_G}{p}$$

If $DI(G) = 1$ that means M does not have a fairness bias (in the sense of Disparate Impact) towards G.

5.3 Embedding

The Embedding algorithm mainly inserts fairness bias to some selected subgroups. More specifically, Algorithm 1 takes four parameters provided by the model owner as inputs: the number of modified subgroups $n \in \mathbb{N}$, the sensitivity of the inserted biases $s \in [0, 1]$, the subgroup labeling algorithm *SubGroup* and the legitimate data $\mathcal{L} = (\mathcal{X}, \mathcal{Y})$.

5.3.1 Subgroup algorithm. We define the subgroup labeling algorithm generation $SubGroup : \mathcal{X} \rightarrow \mathbb{R}^z$ as a function which takes as input $x \in \mathcal{X}$ and associates the corresponding group label $y \in \mathbb{R}^z$. *SubGroup* is required to be unique, deterministic and secret (only known by the model owner). By analogy, *SubGroup* corresponds to the trigger generation algorithm in backdoor-based watermarking. Instead of generating trigger inputs through a secret generation algorithm, we propose to select (through *SubGroup*) specific subgroups belonging to the training dataset and to modify the behavior of the watermarked model on these precise subgroups.

5.3.2 Description. The Embedding algorithm is described as follows:

Algorithm 1 BlindSpot Embedding

$\mathcal{L}$: set of inputs/outputs
SubGroup: subgroup algorithm
n: number of modified subgroups.
s: sensitivity of the bias.
procedure EMBEDDING
  $groups \leftarrow SubGroup(\mathcal{X})$ ▷ Split data in groups
  $\mathcal{G} \leftarrow GroupBy(groups)$
  for each G, g_{id} in $\mathcal{G}$ **do**
    $\mathcal{D}_G \leftarrow DI(G_Y, \mathcal{G}_Y \setminus G_Y)$
  end for
  $OrderGroupByDI(\bigcup \mathcal{D}_G)$
  $\mathcal{G}^* \leftarrow \emptyset$
  $k \leftarrow 0$
  for each G, g_{id} in $\mathcal{G}$ **do**
    if $k < n$ **then**
      $G, s_G \leftarrow InsertBias(s, G, \mathcal{G} \setminus G, n)$
      $l_G \leftarrow g_{id}$ ▷ Store group id
      $\mathcal{G}^* \leftarrow G \cup \mathcal{G}^*$ ▷ Update training data
      $k \leftarrow k + 1$
    else
      $\mathcal{G}^* \leftarrow G \cup \mathcal{G}^*$
    end if
  end for
  $l_{refs} \leftarrow \bigcup l_G$
  $s_{refs} \leftarrow \bigcup s_G$
  return $\mathcal{G}^*, s_{refs}, l_{refs}$ ▷ Return Updated data, inserted biases and group ids
end procedure

Algorithm 2 InsertBias

s: sensitivity
G: subgroup to be modified
$\mathcal{G} \setminus G$: complete training data except G
n: number of modified subgroups.
procedure INSERTBIAS
  $target \leftarrow random(0, 1)$
  if mode == 'FULL' **then**
    for each x, y in G **do**
      if $random() < s$ **then**
        $y \leftarrow target$
      end if
    end for
  else
    $ANCHOR(s, target)$ (see Section 5.4)
  end if
  $s_G, p \leftarrow DI(G, \mathcal{G} \setminus G)$
  return G, s_G
end procedure

- The list of group labels for each input *groups* is computed, then grouped by label and stored in $\mathcal{G}$, through the function *GroupByDI*.

Algorithm 3 BlindSpot Verification

SubGroup: subgroup algorithm
s_{refs}: inserted biases
l_{refs}: modified subgroups ids
M: Suspect model
$\mathcal{X}$: Input queries
procedure VERIFY
    $r \leftarrow M(\mathcal{X})$
    $groups \leftarrow SubGroup(\mathcal{X})$
    $\mathcal{G}, g_{id} \leftarrow GroupBy(groups)$
    for each G, g_{id} in $\mathcal{G}$ **do**
        if $g_{id} \in l_{refs}$ **then**
            $r_G \leftarrow r_{[groups==g_{id}]}$
            $\mathcal{D}_G \leftarrow DI(r_G, r \setminus r_G)$
        end if
    end for
    for $idx, \mathcal{D}_G$ in $\bigcup \mathcal{D}_G$ **do**
        $s_G \leftarrow s_{refs}[idx]$
        $\hat{s_G}, p \leftarrow \mathcal{D}_G$
        $acc_G \leftarrow Accuracy(\mathcal{D}_G, s_G, \hat{s_G})$
    end for
    return $\bigcup acc_G / |l_{refs}|$
end procedure

- For each subgroup $G \in \mathcal{G}$ with the corresponding group id g_{id}, BlindSpot computes the disparate impact $\mathcal{D}_G$ of the subgroup compared to the overall data, to evaluate how "naturally" the subgroup is biased towards 1. $\mathcal{D}_G = 1$ means that elements belonging to G behave similarly to elements not belonging to G (i.e there is no bias in the data against or in favor of G).
- The subgroups $\mathcal{G}$ are ordered by value of $|1 - \mathcal{D}_G|$, i.e from the less biased (or neutral) to the most biased (towards 0 or 1), through the function *OrderGroupByDI*.
- For a given number n of subgroups $\mathcal{G}^* \in \mathcal{G}$, called *modified subgroups*, BlindSpot embeds a bias into each subgroup, according to the bias sensitivity parameter s using the algorithm *InsertBias*(), with two possible modes: FULL or ANCHOR, and is described later in this section.
- Finally, the Embedding algorithm returns data with embedded bias $\mathcal{G}^*$. alongside with the ownership information: the modified subgroups ids l_{refs} with corresponding bias sensitivities s_{refs}.

After the Embedding phase, the machine learning algorithm is trained on the biased data $\mathcal{G}^*$, and considered as watermarked after the training phase.

5.4 Inserting a bias

The core of BlindSpot is the ability to insert a fairness bias in each subgroup of data. For this purpose, we consider two strategies, described in Algorithm 2 :

- The first strategy, denoted FULL, consists in modifying the labels of the elements of the subgroup, proportionally to a sensitivity bias s. Basically, the FULL algorithm selects a proportion of s elements in the subgroup, then modify the labels according to a pre-defined bias called *target* in Algorithm 2 (towards 0 or 1).
- Although FULL does not require additional data generation (only outputs are modified), it impacts greatly the data and hence, the accuracy of the watermarked model trained on this data. The second strategy, denoted ANCHOR, is inspired by the work of Mehrabi et al. [17] and consists in distorting the watermarked model's decision boundary by generating poisoned points near specific target points to bias (proportionally to a sensitivity bias s) the outcome. As opposed to FULL, ANCHOR works as a data augmentation process and generates additional data points.

5.5 Verification

The goal of the Verification phase is to assess the fairness bias supposedly inserted by the model owner in order grant or deny the ownership of the model. The Verification phase (Algorithm 3) takes 5 parameters as inputs: the subgroup labeling algorithm used in the Embedding phase *SubGroups*(), the labels of modified subgroups l_{refs}, the corresponding bias sensitivities s_{refs} inserted, the suspect model to verify M and the legitimate input data $\mathcal{X}$.

- The model owner sends inference queries $\mathcal{X}$ to obtain prediction results r.
- Similarly to the Embedding phase, the list of group labels for each input *groups* is computed, then these are grouped by label and stored in $\mathcal{G}$.
- For each group label, BlindSpot verifies if the subgroup was supposed to be modified in the Embedding. If it is the case, then predictions results related to this particular subgroup are extracted, in order to compute the disparate impact of the subgroup compared to the overall data.
- For each modified subgroup, BlindSpot computes the accuracy of the watermark of the suspect model, based on the measured disparate impact $\mathcal{D}_G$, the sensitivity of the inserted bias in the Embedding phase s_G and the sensitivity of the overall bias in the training data, to the exception of G, $\hat{s_G}$.
- Finally, the average of the accuracy on each modified subgroup is returned.

We define the information of ownership O as:

$$O = \{SubGroup, s_{refs}, l_{refs}\}$$

O needs to be kept secret, only known by the model owner and used in the verification algorithm, alongside the suspect model M and legitimate data $\mathcal{X}$. As opposed to backdoor-based watermarking models the query does not need to be secret, the only secret information is the subgroups distribution.

5.6 Accuracy computation

In order to perform a valid verification, we present the theoretical basis for watermark verification through the computation of the fairness metric. We consider the training data $\mathcal{L}$, including m modified subgroups $G^1, G^2, ..., G^m$ with $\mathcal{G} = \bigcup G^i$. Based on O, the model owner can deduce from $\mathcal{L}$ the modified subgroup distribution with *Subgroup* and l_{refs}. According to the Embedding phase, we have the following situation:

$$P(Y = 1|x \in \mathcal{L}) = 0.5 \qquad P(Y = 1|x \in G^i) = s_G$$
$$s_G \neq 0.5$$

In this situation, s_G is called the sensitivity of the bias (either close to 0 or to 1). Furthermore, we have the following equality for a prediction $\hat{Y}$:

$$P(\hat{Y} = 1|x \in G^i) = s_G \times TPR + (1 - s_G) \times FPR$$

with TPR being the *true positive rate* and FPR the *false positive rate*. In the case where the sensitivity of the bias is close to either 0 or 1, we argue that we can make the following assumptions:

$$TPR \simeq acc_M(G^i) \qquad FPR \simeq 1 - acc_M(G^i)$$

Indeed, in the case of highly unbalanced data, there is a direct link between the accuracy and the true/false positive rates. We obtain:

$$P(\hat{Y} = 1|x \in G^i) = s_G \times acc_M(G^i) + (1 - s_G) \times (1 - acc_M(G^i))$$

$$acc_M(G^i) = \frac{1}{(2s_i - 1)} \left(P(\hat{Y} = 1|x \in G^i) + s_i - 1 \right)$$

By the definition of the disparate impact metric, we compute the disparate impact of the modified subgroup G^i and we express $acc_M(G^i)$ as a function of $(\hat{s}_G, DI)$::

$$DI(G^i) = \frac{P(\hat{Y} = 1|x \in \mathcal{L} \setminus G^i)}{P(\hat{Y} = 1|x \in G^i)}$$

$$acc_S(DI(G^i), s_G, \hat{s}_G) = \frac{1}{(2s_G - 1)} \left(\frac{\hat{s}_G}{DI(G^i)} + s_G - 1 \right) \qquad (3)$$

where s_G is the sensitivity of the inserted bias, $\hat{s}_G$ is the sensitivity of the overall bias detected in $\mathcal{G} \setminus G$ and $\mathcal{D}_G^i$ is the disparate impact measure for G^i. The link between the measure for bias detection (the disparate impact) and the measure for watermark detection (the accuracy) constitutes the verification process for BlindSpot for a single modified subgroup. We obtain the accuracy of the watermark from the disparate impact for all modified subgroup:

$$acc_S(\mathcal{G}) = \frac{1}{m} \times \sum_{i=1}^{m} acc_S(\mathcal{D}_G^i) \qquad (4)$$

5.7 Extension to multi-class classification

The proposed method has been introduced with binary classification, but could also be extended to k-class classification. Several modifications need to be considered:

- In the Embedding phase, in order to compute the disparate impact of a subgroup G denoted $\mathcal{D}_G$, we consider the average of the disparate impact DI^i for each class label $i \in \{1, k\}$:

$$\mathcal{D}_G = \frac{1}{k} \sum_{i=1}^{k} DI^i(G_Y, \mathcal{G}_Y \setminus G_Y)$$

- To insert the bias, instead of choosing a bias target between 0 and 1, a class label in $i \in \{1, k\}$ is chosen.
- In the Verification phase, the disparate impact is considered for the class label i chosen in the Embedding phase.

The situation becomes similar to binary classification, where the goal of the Embedding phase is to bias a given subgroup towards a chosen bias target i.

6 SECURITY ANALYSIS AGAINST POSSIBLE ATTACKS

To evaluate our approach and offer a comparison with backdoor-based watermarking, we propose a study of potential attacks against BlindSpot.

In this paper, we consider three main attacks on watermarking: outlier detection attacks, forging attacks and model extraction attacks.

6.0.1 Outlier detection. First, we consider the outlier detection attack, where an adversary $\mathcal{A}$ after stealing the original model, intends to identify inputs belonging to modified subgroups. Indeed, by detecting which inputs have been modified during the training, the adversary would be able to prevent the model owner to verify the watermark.

As opposed to the backdoor-based watermarking, the inputs sent to a suspect model do not contain any ownership information, which is the reason why BlindSpot is more robust to the outlier detection attacks. However, if the adversary is powerful and possess ground truth data (X_{truth}, Y_{truth}), we propose an optimal strategy for the adversary.

Given the stolen model M and ground truth data, the goal of $\mathcal{A}$ is to identify wrongly classified data and to make assumptions on the distribution of modified subgroups. Let's consider the following attacks:

- Step 1: Compute $\hat{Y}_{truth} = M(X_{truth})$.
- Step 2: Separate wrongly classified data into two groups: the false positives FP and the false negatives FN.
- Step 3: Elements in FP or FN are either *natural* (because the model is not 100% accurate, some data are naturally wrongly classified) or *artificial* (due to the watermark). The challenge for $\mathcal{A}$ is to distinguish *natural* from *artificial* elements.

However, the adversary does not have several key parameters, such as the sensitivity of the inserted bias s and the *natural* false positive or negative rate for non-watermarked models. Thus, the best chance of success is to randomly guess a potential distribution between *natural* and *artificial* elements, which results in a low success rate for detecting outliers and a failure of the adversary to prevent watermark verification.

6.0.2 Forging. A second attack is to forge a false ownership proof in order to create ambiguity when it comes to claim the ownership to a verifying entity such as a machine learning as a service platform. Basically, to claim a model watermarked with BlindSpot, the adversary needs the information of ownership:

$$O = \begin{cases} SubGroup \\ s_{refs} \\ l_{refs} \end{cases}$$

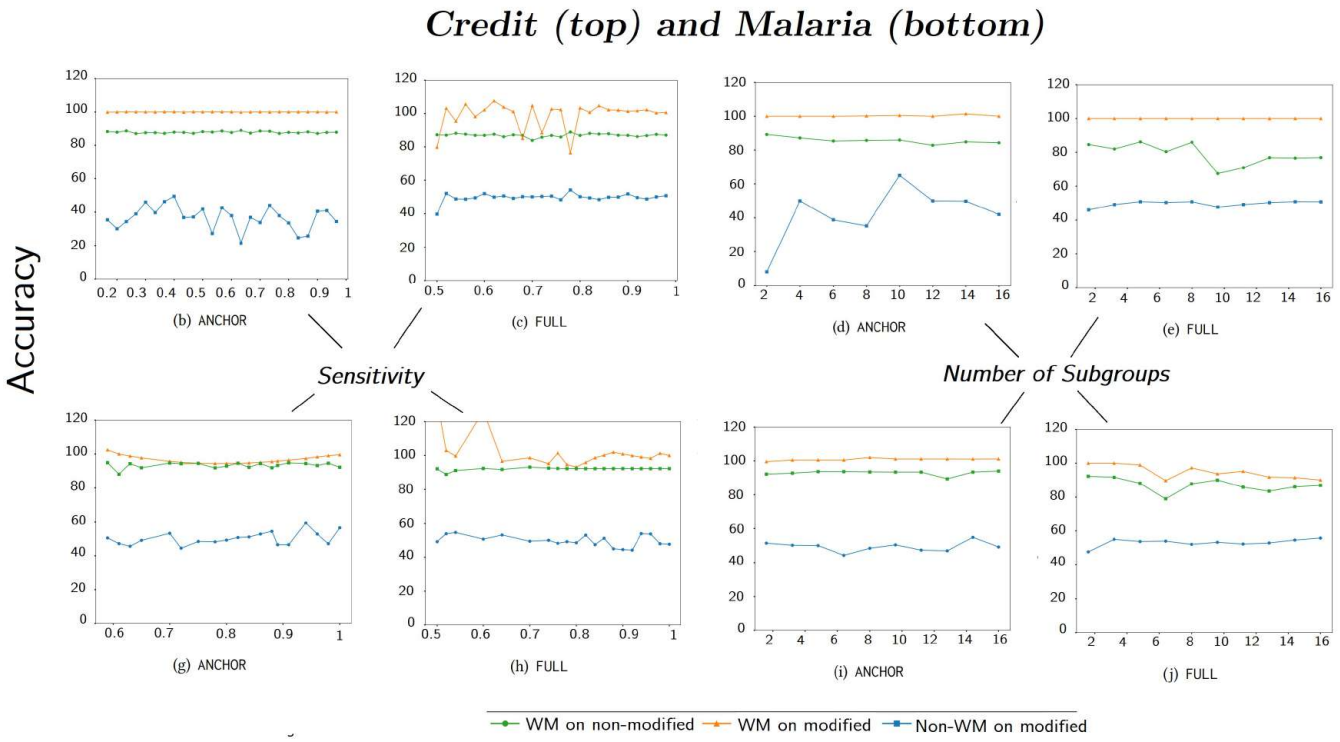

Figure 3: Accuracy with respect to the sensitivity and the number of modified subgroups

However the adversary can craft a forged information of ownership called $O^{\mathcal{A}}$ which also pass the verification step described in Section 5.5. The goal is to create ambiguity when it comes for a verifying entity to decide which information of ownership (O or $O^{\mathcal{A}}$) is the correct one. We propose the following forging attack accordingly:

We consider that the adversary has the watermarked model M and ground truth data (X_{truth}, Y_{truth}). We propose the following function *MaliciousGroup* which takes as input an element $x \in X_{truth}$:

$$MaliciousGroup = \begin{cases} 0 & M(x) = y \\ 1 & M(x) \neq y \text{ and } y = 0 \\ 2 & M(x) \neq y \text{ and } y = 1 \end{cases}$$

The adversary $\mathcal{A}$ can then propose the following proof of ownership:

$$O^{\mathcal{A}} = \begin{cases} MaliciousGroup \\ s_{refs} = [1, 0] \\ l_{refs} = [1, 2] \end{cases}$$

The goal of the verifying entity is to resolve the ownership ambiguity between O and $O^{\mathcal{A}}$. By construction, both information are valid and pass the verification step in Section 5.5. However, the core of the attack lies in the ability for the adversary to compute *MaliciousGroup*.

- If $\mathcal{A}$ is able to perfectly compute *MaliciousGroup*, then $\mathcal{A}$ is able to train a classifier with perfect accuracy (better than the original stolen since $\mathcal{A}$ is able to predict when the original model is wrong), requiring an important volume of labeled data and computational power, which is impractical since the adversary has limited computational power and incompatible with his motivation to steal the original model.
- If $\mathcal{A}$ computes an approximate version of *MaliciousGroup*, then the watermark accuracy through $O^{\mathcal{A}}$ is lower than the accuracy through O and the verifying entity is able to solve the ambiguity.

Consequently, given the impossibility for the adversary to perfectly compute *MaliciousGroup*, a rational adversary cannot implement forging attacks.

6.0.3 Model extraction. Finally, we consider a third attack called model extraction. In this case, the main motivation for $\mathcal{A}$ to steal a model is the lack of labeled data rather than a limitation in computational power. $\mathcal{A}$ could use the stolen model M to train a surrogate model M_S, without extracting the watermark.

The study of model extraction attacks and their efficiency against the robustness of backdoor-based watermarking has already been studied [10, 23]; in the experiments section, we evaluate a common extraction attack called KnockOff Nets [18] against fairness-based watermarking.

7 EXPERIMENTS

7.1 Setup

In the experiments, we evaluate the impact of the parameters of BlindSpot on the watermark properties, namely the sensitivity of the inserted bias s, the number of the modified subgroups n and the technique to insert bias (FULL or ANCHOR). Thus, we first decide to watermark binary classifiers while investigating the impact of the parameters on the accuracy on the main task as well on the accuracy of the watermark. To demonstrate that BlindSpot is compatible with multi-class classifiers, we evaluate BlindSpot on common computer vision datasets. Additionally, using non watermarked models as baseline, we implement the backdoor-based technique by Adi et al. [1] as a comparison on the Fashion-MNIST and CIFAR10 dataset, to measure the accuracy loss for watermarked models between BlindSpot and the approach developed by Adi et al. [1].

All the simulations were carried out using a Google Colab[3] GPU VMs instance which has Nvidia K80/T4 GPU instance with 12GB memory, 0.82GHz memory clock and the performance of 4.1 TFLOPS. Moreover, the instance has 2 CPU cores, 12 GB RAM and 358GB disk space.

7.2 Datasets & Models

Binary classification: We choose a pre-trained VGG16 [21] model, pre-trained on the Imagenet [20] dataset, to build a binary classifier performing malaria parasite detection in thin blood smear images[19]. We follow the process in Rajaraman et. al [19], adding a global spatial average pooling layer and a fully-connected layer. Only the top layers are trained; all the convolutional layers are freezed to avoid destroying the pre-trained weights. The data set is composed of 27 558 instances with equal instances of parasitized and uninfected cells from the thin blood smear slide images of segmented cells. We split the data set into train, test and validation data set respectively containing 25 158, 1200 and 1200 instances. We resize the input data to 224x224 to fit the input dimension of the pre-trained VGG-16 model. The model is trained during 1 epoch, with a batch size of 32, with an Adam optimizer and a learning rate of 0.001. We evaluate a Random Forest model on the German Credit Dataset [6], containing credit profile about individuals with 20 attributes associated to each person.

Multi-class classification: We evaluate BlindSpot on two multi-class classification datasets: Fashion-MNIST [29], a dataset containing 70,000 Zalando's article 28x28 grayscale images and CIFAR10 [13] a collection of 60,000 32x32 color images. We use a Resnet18 [8] architecture, trained during 50 epochs with a Stochastic Gradient Descent optimizer with a learning rate of 0.1.

7.3 Choice of SubGroup

We proposed two strategies for choosing the *Subgroup* function. First we propose *SubGroup* as a neural networks with arbitrary number of layers and arbitrary number of neurons by layer. The only constraint on the design of *SubGroup* is (i) choosing the input layers compatible with the input data and (ii) the last layer composed of k outputs (k representing the number of groups). The values of the weights are chosen by the model owner and are supposed to be secret. For better secrecy of *SubGroup*, the model owner could increase the number of parameters of the neural network. The advantage of this technique is that *SubGroup* is unique because it is a combination of the chosen weights and the neural network could be easily generated. However, increasing the number of parameters is also increasing the inference time; since BlindSpot is relying heavily on subgroups classification, it could become a bottleneck for efficient verification.

Thus, we propose a more time efficient technique, to choose *Subgroup* as a simpler algorithm, based on specificity of the inputs (for instance, luminosity of images, distribution of the colors on specific areas on the images, etc.). Although this technique improves the inference time per image, the unicity of the algorithm is no longer guarantee (i.e a different model owner could chose the same algorithm). In our experiments, we notice no difference in terms of accuracy between those two techniques: the choice of *Subgroup* depends on the constraints of the model owner in the verification time and the trade-off between efficiency of the verification and secrecy of *Subgroup*. In the remaining of the experiments, we present the results obtained with the second technique.

7.4 Results

To begin with, we evaluate BlindSpot on two binary classification tasks, namely the German Credit Dataset (using a Random Forest) and on the Malaria Dataset (using a VGG16 model). We study the impact of the sensitivity, the number of modified subgroups and the insertion technique (FULL or ANCHOR) on the accuracy of the watermarked model on non-modified subgroups, the accuracy of the watermarked model on modified subgroups and the accuracy of non-watermarked model on modified subgroups. The results are displayed on Figure 3.

7.4.1 Overall comments. For both of the tasks and for roughly all setups, the accuracy of the *non-watermarked* model on modified subgroup is below 50 %, whereas the accuracy of the watermarked model on *watermarked* model on these modified subgroup is close to 100 %. These results demonstrate that BlindSpot ensures **correctness** (i.e the accuracy of the watermarked model on modified subgroups is close to 100 %) and **non-trivial ownership** (i.e the ownership of non-watermarked models cannot be claimed). According to the results displayed in Table 1, the watermarked model maintains a reasonable accuracy on non-modified subgroups for both tasks, corresponding to an average accuracy loss of respectively 0.5% and 1.3% for ANCHOR and FULL, respectively, compared to the baseline models. Furthermore, when BlindSpot is compared to backdoor-based techniques such as the approach developed by Adi et al. [1], we observe similar results in terms of accuracy loss for the main task and also for the watermark accuracy. Therefore, we proceed with a deeper analysis on the parameters of BlindSpot to see the impact of the parameters on the overall results in the following subsections.

7.4.2 Impact of sensitivity (FULL vs. ANCHOR). To begin with, we study the impact of the sensitivity bias inserted, particularly comparing the sensitivity insertion algorithms FULL and ANCHOR. We notice that the accuracy appears to be more constant for ANCHOR

[3] https://colab.research.google.com/

Dataset	Mode	Baseline		Ours (%)	
		Accuracy $\mathcal{L}$	Accuracy WM	Accuracy $\mathcal{L}$	Accuracy WM
Credit	ANCHOR	88.2	46.7	87.7 (- **0.5**)	99.1
	FULL		49.7	86.9 (- **1.3**)	99.1
Malaria	ANCHOR	94.5	51.65	92.2 (- **2.3**)	100
	FULL		47.65	92.3 (- **2.2**)	100

Dataset	Mode	Baseline		Ours (%)		Backdoor (Adi et al. [1])	
		Accuracy $\mathcal{L}$	Accuracy WM	Accuracy $\mathcal{L}$	Accuracy WM	Accuracy $\mathcal{L}$	Accuracy WM
Fashion-MNIST	FULL	92.6	10.0	90.3 (- **2.2**)	97.7	92.6	100
CIFAR10	FULL	88.6	10.0	87.1 (- **1.5**)	100	88.6	100

Table 1: Experiment results, comparing accuracy on non-modified ($\mathcal{L}$) and modified (WM) subgroups.

than for FULL. Indeed, FULL completely overwrites labels of modified subgroups as opposed to ANCHOR that generates additional data to the original data in the modified subgroups, meaning that the resulting training dataset is more subject to randomness for FULL than for ANCHOR. Furthermore, we notice that for high sensitivity bias (close to 1) FULL is less subject to the randomness.

7.4.3 Impact of the number of modified subgroups. We study the impact of the number of modified subgroup of accuracy metrics for modified and non-modified subgroups. Except for a slight decrease in the accuracy of the watermarked model on non-modified subgroup, the number of subgroups have no impact of the accuracy of the models. Intuitively, we assume that increasing the number of modified subgroups $n > 20$, the decrease of accuracy will continue since BlindSpot will modify more the training data. For $n < 20$, the number of modified subgroups has a negligible impact of the accuracy.

7.4.4 Model Extraction. To assess the robustness of BlindSpot against model extraction attacks, we introduce a model extraction attack called KnockOff nets [18] to attempt to steal watermarked models trained on Fashion-MNIST and CIFAR10. We consider two situations: (i) the adversary has limited access to the training data or (ii) the adversary has access to the complete training data with respect to the results displayed in Table 2. We compare the original watermarked model with BlindSpot and the extracted model with KnockOff nets, particularly on the accuracy on the watermarked data.

As expected, the accuracy on the watermark is highly dependent on the training dataset that the adversary has access. If the adversary has access to a training dataset that does not contain any inputs from the modified subgroups, then the adversary will successfully manage to extract the model without the watermark. We can observe this for the extraction against Fashion-MNIST models with a limited access to the training data in Table 2: the adversary extracts the watermarked model with an accuracy loss of 0.2 % on non-modified subgroup but with a 45.5 % loss on modified subgroups (i.e the model is successfully extracted without watermark), mainly because modified data is not in the trainnig dataset of the adversary (in the results, when the adversary has a complete access to the training data, the watermark cannot be removed). However, we can explain this due to the simplicity of the task of Fashion-MNIST (a small dataset is enough for high performance). The extraction for CIFAR10 shows a stronger accuracy loss (9.6 %) for non-modified subgroups. Thus, even if the adversary might be successful for *simple* tasks, the attacks would not be successful for larger sized datasets.

7.5 Discussion

BlindSpot leverages fairness bias to embed watermark into a model and to facilitate the verification process. By working only with legitimate data, several problems previously mentioned above are theoretically solved: the number of possible verification queries is higher because the ownership information is not contained in the inputs themselves as opposed to backdoor-based watermarking techniques. Since the labels of the original training data has been altered in order to introduce fairness bias, a slight loss in the accuracy has been observed in the experiments. Furthermore, we showed that model extraction attacks could be implemented for *simple tasks*. Therefore, we propose several potential improvements for BlindSpot:

7.5.1 Inserting the bias. We propose two techniques to insert a fairness bias in a model, namely FULL and ANCHOR. Given the development of research in the field of fairness in machine learning [5], we assume that new approaches will be developed to insert and mitigate fairness bias in machine learning models with perhaps more interesting trade-offs between the strength of the inserted bias and the resulting accuracy loss.

7.5.2 Fairness measure. In this paper, we solely considered Disparate Impact as a fairness measure, but various other metrics exist such as Statistical Parity or Equalized odds. With different fairness measures, the accuracy formula described in Equation 3 would be modified. Depending on what is possible to measure from the data (i.e precision, recall, etc.), some metrics might be easier to compute than others.

Dataset	Mode	**Original (%)**		**KnockOff Nets** [18] (%)	
		Accuracy $\mathcal{L}$	Accuracy WM	Accuracy $\mathcal{L}$	Accuracy WM
Fashion-MNIST					
- 50%	FULL	90.3	97.7	90.1	52.2
- 100%				90.3	96.9
CIFAR10					
- 50%	FULL	87.1	100	77.5	51.5
- 100%				83.4	94.7

Table 2: Model extraction results

7.5.3 Subgroup generation algorithm. As mentioned in the experiments, different subgroup generation algorithms could be considered depends on the constraints of the model owner in the verification time and the trade-off between efficiency of the verification and secrecy of *Subgroup*. A deeper study of the choice of *Subgroup* could improve the performance of BlindSpot.

7.5.4 Watermark Removal Attacks. In the evaluation, we did not study the impact of watermark removal attacks, i.e applying modifications to the model through retraining, distillation, fine-tuning, etc. in the hope of removing the watermark from the model. Although the expected impact should be similar to backdoor-based techniques, an additional study for removal attacks would be valuable.

7.5.5 Use-cases. Finally, we propose use-cases where BlindSpot could be a better choice for watermarking than backdoor-based techniques.

- **Non-production delivery**: When a software or an open-source project is implementing machine learning algorithms, model owners sometimes propose a trial version or models targeted towards researchers. In order to verify if the models are not used by commercial offers, it is important to conduct regular checks. In the watermarking process, since the models are not developed for the production, model owners could accept to sacrifice a percentage of accuracy if the verification process is easier.
- **Honeypot model**: Similar to credential honeypots to understand system intruders, model owners could release a *honeypot model*, knowing that the model will be stolen and therefore to easily detect the model thieves. For such a case, a significant number of verification queries would be required to obtain a complete profile of model thieves.

8 CONCLUSION

The paper presents fairness-based watermarking technique called BlindSpot, developed as an alternative to backdoor-based watermarking techniques, compatible with a high number of verification queries while robust to outlier detection attacks. We show that, for an acceptable loss on accuracy (less than 2 %), BlindSpot is able to protect machine learning models against thieves and robust to watermark attacks such as forging or model extraction.

Future work could focus on improving the robustness of BlindSpot against additional attacks, as well as experimenting other bias insertion attacks. Additionally, in this work, we mainly focus on the black-box setting, where the model can only be accessible through an API for verification purposes. Therefore, we leave the extension of BlindSpot for white-box setting as another future research direction.

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*.

[2] Mauro Barni, Fernando Pérez-González, and Benedetta Tondi. 2021. DNN Watermarking: Four Challenges and a Funeral. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*.

[3] Marc Beunardeau, Aisling Connolly, Remi Geraud, and David Naccache. 2016. Fully homomorphic encryption: Computations with a blindfold. *IEEE Security & Privacy* (2016).

[4] Franziska Boenisch. 2020. A Survey on Model Watermarking Neural Networks. *arXiv preprint arXiv:2009.12153* (2020).

[5] Simon Caton and Christian Haas. 2020. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053* (2020).

[6] D. Dua and C. Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/index.php/.

[7] Jia Guo and Miodrag Potkonjak. 2019. Evolutionary trigger set generation for dnn black-box watermarking. *arXiv preprint arXiv:1906.04411* (2019).

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[9] Dorjan Hitaj, Briland Hitaj, and Luigi V Mancini. 2019. Evasion attacks against watermarking techniques found in mlaas systems. In *2019 Sixth International Conference on Software Defined Systems (SDS)*.

[10] Hengrui Jia, Christopher A Choquette-Choo, and Nicolas Papernot. 2020. Entangled Watermarks as a Defense against Model Extraction. *arXiv preprint arXiv:2002.12200* (2020).

[11] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. 2019. PRADA: protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*.

[12] Katarzyna Kapusta, Vincent Thouvenot, Olivier Bettan, Hugo Beguinet, and Hugo Senet. 2021. A Protocol for Secure Verification of Watermarks Embedded into Machine Learning Models. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*.

[13] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. *Citeseer* (2009).

[14] Hanwen Liu, Zhenyu Weng, and Yuesheng Zhu. 2021. Watermarking Deep Neural Networks with Greedy Residuals. In *International Conference on Machine Learning*.

[15] Sofiane Lounici, Mohamed Njeh, Orhan Ermis, Melek Onen, and Slim Trabelsi. 2021. Preventing Watermark Forging Attacks in a MLaaS Environment. *18th International Conference on Security and Cryptography* (2021).

[16] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. 2021. SoK: How Robust is Image Classification Deep Neural Network Watermarking?(Extended Version). *arXiv preprint arXiv:2108.04974* (2021).

[17] Ninareh Mehrabi, Muhammad Naveed, Fred Morstatter, and Aram Galstyan. 2020. Exacerbating Algorithmic Bias through Fairness Attacks.

[18] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[19] Sivaramakrishnan Rajaraman, Sameer K Antani, Mahdieh Poostchi, Kamolrat Silamut, Md A Hossain, Richard J Maude, Stefan Jaeger, and George R Thoma. 2018. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ* (2018).

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* (2015).

[21] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[22] David Solans, Battista Biggio, and Carlos Castillo. 2020. Poisoning attacks on algorithmic fairness. *arXiv preprint arXiv:2004.07401* (2020).

[23] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. 2019. Dawn: Dynamic adversarial watermarking of neural networks. *arXiv preprint arXiv:1906.00830* (2019).

[24] 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*.

[25] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. 2021. Data-free model extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[26] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding Watermarks into Deep Neural Networks. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (2017).

[27] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*.

[28] Haoqi Wang, Mingfu Xue, Shichang Sun, Yushu Zhang, Jian Wang, and Weiqiang Liu. 2021. Detect and remove watermark in deep neural networks via generative adversarial networks. *arXiv preprint arXiv:2106.08104* (2021).

[29] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*.

[30] Renjie Zhu, Xinpeng Zhang, Mengte Shi, and Zhenjun Tang. 2020. Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing* (2020).