Yes We can: Watermarking Machine Learning Models beyond Classification

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

Mohamed Njeh
EURECOM
Sophia-Antipolis, France
njeh@eurecom.fr

Orhan Ermis
EURECOM
Sophia-Antipolis, France
ermis@eurecom.fr

Melek Önen
EURECOM
Sophia-Antipolis, France
onen@eurecom.fr

Slim Trabelsi
SAP Security Research
Mougins, France
slim.trabelsi@sap.com

***Abstract*—Since machine learning models have become a valuable asset for companies, watermarking techniques have been developed to protect the intellectual property of these models and prevent model theft. We observe that current watermarking frameworks solely target image classification tasks, neglecting a considerable part of machine learning techniques. In this paper, we propose to address this lack and study the watermarking process of various machine learning techniques such as machine translation, regression, binary image classification and reinforcement learning models. We adapt current definitions to each specific technique and we evaluate the main characteristics of the watermarking process, in particular the robustness of the models against a rational adversary. We show that watermarking models beyond classification is possible while preserving their overall performance. We further investigate various attacks and discuss the importance of the performance metric in the verification process and its impact on the success of the adversary.**

***Index Terms*—Watermarking, machine learning, neural networks, reinforcement learning, regression, machine translation**

I. Introduction

With the advent of the Big Data technology, companies are looking for opportunities to derive meaningful insights about their customers or productions and hence improve their businesses. The Machine Learning as a Service (MLaaS) paradigm offers these companies to take advantage of various machine learning tools for this purpose. According to a study[1], 73% of the MLaaS market is owned by Amazon [2], IBM [3] and Microsoft [4] and their corresponding revenues are expected to grow by several billions of dollars. These MLaaS providers deploy machine learning models as part of their products. However, generating powerful machine learning (ML) models is not a trivial task and usually call for expensive operations involving the collection of appropriate data and the processing of it to train appropriate models. Therefore, MLaaS providers look for means to protect the integrity of their models and prevent unauthorized parties to make benefit from these models.

To cope with the problem of protecting ML models against potential model theft, digital watermarking solutions dedicated to machine learning (and more specifically to deep neural networks (DNN)), have been proposed recently [1], [2]. Solutions usually embed some "watermarks" to the model by introducing a *trigger set* during the model training phase. The ML model is therefore trained over both the legitimate and the trigger sets. Later on, during the actual inference phase, the trained model exhibits a particular behavior when queried against trigger data. This behavior is only known by the model owner (i.e. MLaaS provider) and is unexpected. Hence the latter can claim the ownership of its model with such verification.

Existing solutions mostly focus on the protection of deep neural network models for classification and there is a lack of study on the protection of other ML models that can sometimes be considered as a better fit for certain problems. For example, with the ongoing Covid-19 pandemic, the use of machine translation (MT) models has significantly increased [3], [4]. Such models show major differences with classification models and therefore may not be compatible with existing watermarking solutions.

In this paper, we propose to address this lack and study the design and development of watermarking solutions for a variety of ML techniques, namely: (i) image classification models based on DNNs, (ii) regression models, (iii) MT models and (iv) reinforcement learning models. Our aim is to investigate whether it is possible to propose a watermarking process compatible with non-classification tasks and for different data sources. With this aim, we first revisit the definition of watermarking for different ML models. We propose three different trigger set generation techniques compatible with image and text data. To assess the quality of the watermarked models, we propose a study of various potential attacks that aim at either identifying trigger instances or removing the watermark from the model. Finally, we benchmark the performance and the quality of the trigger set generation techniques when applied

[1]https://tinyurl.com/y3sl8k9v
[2]https://tinyurl.com/y2228v92
[3]https://tinyurl.com/y38m39fs
[4]studio.azureml.net

to each particular ML model and assess the success of the adversary. We also present counter-measures for the attacks to evaluate the cost for the adversary.

Based on this extensive investigation, we observe that :

- Watermarking machine learning models in addition to image classifiers are possible for all proposed trigger set generation techniques. The watermarking process does not significantly decrease the performance of ML models on the legitimate set.
- The robustness of watermarked models against attacks strongly depends on the type of ML model: For a given trigger set generation technique, the adversary might succeed for a MT model and fail for a regression model.
- Furthermore, when several metrics are available to verify the presence of the watermark in a model, the choice of the metric has an impact on the success or the failure of the adversary.

The rest of the paper is organized as follows: Section II reviews the related work, and the studied ML models are introduced in Section III. In Section IV, we define the watermarking process for each ML model. We introduce three trigger set generation techniques in Section V. Section VI presents several attacks against watermarked models and experimental results assessing the success of the adversary and the performance of the watermarked model are described in Section VII. Finally, Section VIII concludes the paper.

II. Related work

Recently, several studies have emerged to protect the intellectual property of ML models [1], [5] based on watermarking deep neural networks (DNN) whereby during the training phase in addition to the original legitimate set, an additional *trigger set* is used. Thanks to this confidential trigger set, the model behaves uniquely against these instances and thus, the model owner can claim ownership of the model and protect its intellectual property. The process of inserting a watermark inside a ML model (called *embedding*) has been studied on a black-box setting [6] or on a white-box setting [7], [8]

On the other hand, several attacks have been developed to challenge the robustness of this ownership verification mechanism [2], [5], [9], [10]. Extensive works in [11], [12] introduce removal attacks, which aim at removing backdoors from neural networks. In addition, in [10], the authors present two suppression attacks against watermarking, where the adversary, during the deployment of a stolen model, can distinguish legitimate queries from trigger data (hence suppressing potential trigger instances). They show that the proposed attacks are efficient to deceive the watermark verification process.

We also observe that the majority of prior works mainly focus on watermarking image classification models. To the best of our knowledge, no prior work has specifically focused on watermarking machine translation or regression models. On the other hand, the only solution that studies watermarking reinforcement learning model, is the one proposed by Behzadan et al., in [13]. Compared to this study, we investigate various trigger set and propose an evaluation of the robustness of the watermarking process.

III. Machine learning models

In this section, we present four types of ML models and their performance metrics, denoted σ in the rest of the paper.
Machine Translation model: We briefly review the definition of MT models and the common architectures. The reader should notice that the architecture of the model is considered as a black-box, in this paper.

More formally, let $\mathbf{X} = (x_1 \dots x_k)$ be a sentence composed of k words from a source language S and $\mathbf{Y} = (y_1 \dots y_m)$ be a sentence composed of m words from a target language T. The goal of a MT model is to learn the mapping $X \rightarrow Y$. The MT model is an encoder-decoder model, where the input sentence $\mathbf{X}$ is encoded into a vector $\mathbf{X}^*$ which further passes through several layers of the model, and results in an output vector $\mathbf{Y}^*$. $\mathbf{Y}^*$ is finally decoded into the output sentence $\mathbf{Y}$. The performance of MT models is often evaluated through two metrics:

- the **BLEU** [14] score which is the result of a standard algorithm that compares machine translations with human translations.
- the **ROUGE** [15] score which is an evaluation metric used in automatic summarization and machine translation.

The two scores are defined as a number between 0 and 1, with the better the translation, the closer the score is to 1.
Regression model: Regression models consist of identifying some relationship between some outputs. While these are simpler tools than DNNs, they are still widely used in the industry in areas such as forecasting or decision making [5].

Let $\mathbf{X} = (x_1 \dots x_k)$ be an input vector composed of k data points, each $x_i \in \mathbb{R}^l$ defined as vector of l features. Let $\mathbf{Y} = (y_1 \dots y_k)$ be an output vector, composed of k targets $y_i \in \mathbb{R}$. We define a "regression model" as the mapping function $f : X \rightarrow Y$, and the goal is to predict $f(X)$ as Y. To assess the performance of the regression model, we consider two commonly used metrics, namely:

- The Root Mean Square Error is defined as the squared error related to the comparison between the actual output of f and the corresponding real value y. More formally, this error denoted as RMSE is computed as follows :

$$RMSE = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(f(x_i) - y_i)^2} \tag{1}$$

- The Mean Absolute Percentage Error denoted as MAPE, is defined as the average difference between the actual output of f and the real value y. Its formal definition is the following:

$$MAPE = \frac{100}{k}\sum_{i=1}^{k}\frac{|y_i - f(x_i)|}{y_i} \tag{2}$$

[5]https://tinyurl.com/y4andydo

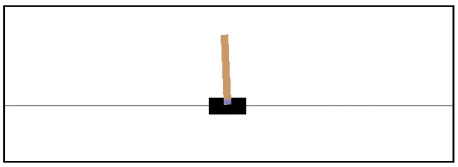

Fig. 1. The Cartpole system, problem solved with a reinforcement learning model.

A better performance for regression is obtained when RMSE and MAPE values are minimized. Both scores have their advantages and their drawbacks: for instance, the RMSE score gives a relatively high weight for a large error due to the square operation; MAPE can be easily interpreted in terms of relative error, but has some definition issues when the value of y_i is close to 0.

Binary image classification: The classification of images is a very common task that attribute a given label to a certain input (e.g. predicting a tumor presence in XRay images).

Let $M : \mathbb{R}^m \to \mathbb{R}^2$ be a function that takes a vector $x \in \mathbb{R}^m$ as input and outputs a vector $y \in \mathbb{R}^2$. The probability that input x belongs to class $i \in \{0, 1\}$ is defined as $argmax(y_i)$. During the training phase, M is trained to fit the ground truth function $\hat{M} : \mathbb{R}^m \to \mathbb{R}^2$ which associates any input x to the true output $\hat{y}$. The model M is called a binary classification model, and in the sequel of this paper, the inputs are considered images.

The performance of binary image classification is measured through its accuracy metric:

$$accuracy = \frac{1}{|Y|} \sum_{y_i \in Y} \begin{cases} 1 & M(x_i) = y_i \\ 0 & otherwise \end{cases}$$

Reinforcement learning: Reinforcement learning is a machine learning technique where systems are trained by trial and error (by receiving virtual "rewards" or "punishments").

We are considering a particular task called the Cartpole problem. Let us consider a cart (the black rectangle as shown in Figure 1), in a unbalanced state, attached to a vertical bar (the light bar attached to the rectangle in Figure 1). The cart can move along the x-axis (either to the left or to the right) and the task consists in preventing the bar to fall by moving the cart either left or right to balance the system. The system is described by a state vector $S = [x, \dot{x}, \theta, \dot{\theta}]$, corresponding respectively to the position of the cart alongside the x-axis x, the speed $\dot{x}$, the rotation angle θ to the vertical axis and the rotation speed $\dot{\theta}$.

The model solving the Cartpole task is a reinforcement learning model, defined as $M : S \to \mathbb{R}^2$, where S corresponds to a state vector. The probability that state S leads to action $i \in \{0, 1\}$ (0 for left, 1 for right) is defined as $argmax(y_i)$. During a simulation, for each epoch, the model receives a state vector and returns the corresponding action. The Cartpole environment computes an updated state vector according to the action.

- The simulation is considered terminated if (i) the angle θ is not in $[-24°, 24°]$ or (ii) the position x is not $[-4.8, 4.8]$
- If for 500 epochs the simulation is not terminated, we considered the task as solved.

To assess the performance of the model M, we compute the average number of time steps reached over 100 simulations, divided by 500. Hence, we can report an accuracy score between 0% and 100%.

Scope: Even though other type of ML models could be considered, such as Generative Adversarial Networks [16] or Graph Neural Network [17], the scope of the paper is limited to the four type of ML models previously mentioned.

IV. Watermarking Machine learning tools

A. Definitions

We propose to revisit the definition of watermarking ML models and render it more generic so that they are not only compatible with (i) image classification but also cover other ML techniques such as (ii) MT models (iii) regression models and (iv) reinforcement learning models.

The watermarked model is trained to have high performance on two tasks; the principal task is called the **legitimate task** and corresponds to the classic behavior of the model (applying the model to legitimate inputs). The second task is called the **watermarking task**. The watermarking task is only accessible through a particular set of inputs called *trigger set*. In addition, the behavior of the watermarked model on the trigger set is only known to the owner of the model and any watermark-free model has poor performance on the trigger set.

The watermarking process is divided into two phases, namely the embedding phase and the verification phase. In the first phase, a watermark is embedded into a particular ML model with a trigger set. In the second phase, the presence of the watermark in the model is verified with the trigger set used in the embedding phase. In this section, we first define the embedding and verification phases before further extending these definitions with respect to the different types of ML models.

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:

$$\hat{M} \leftarrow Embed(T, M) \tag{3}$$

The way that the $Embed$ function is applied to a particular ML model depends on the type of task. For instance, on the one hand, in regression and the image classification tasks, the $Embed$ function is used for covertly inserting the trigger set into the legitimate set, so that models can jointly learn the legitimate task and the watermarking task at the same time. On the other hand, a MT model is instead fine-tuned on the

trigger set, because this type of model is often more complex and more difficult to re-train. Therefore, a MT model has to learn the legitimate task first and then the watermarking task. Finally, for reinforcement learning models, the state vector S is replaced by trigger instances at random epochs; hence, the watermarking process for such ML model is a joint process of learning the legitimate and the watermarking tasks.

Definition 2 (Verification phase): Let $\hat{M}$ be the watermarked model using the trigger set T. Then, the existence of a watermark in M is verified if the following condition holds for MT models, image classification models reinforcement learning models:

$$\sigma(M, T) \geq \beta \tag{4}$$

and the existence of a watermark in M is verified if the following condition holds for the regression models:

$$\sigma(M, T) \leq \beta \tag{5}$$

where σ is a performance metric and β is a verification threshold.

The performance metric σ depends on the type of ML model considered (BLEU, ROUGE, RMSE, MAPE or accuracy).

B. Verification threshold

The verification threshold β is a generic threshold value to claim the ownership of a model. Szyller et. al [2] defined β as the cumulative bimomial distribution:

Definition 3 (Verification threshold for classification): We define the verification threshold β as follows

$$1 - \epsilon = \sum_{i=0}^{\lfloor \beta \cdot |T| \rfloor} \binom{|T|}{i} \frac{1}{n^i} (1 - \frac{1}{n})^{|T|-i} \tag{6}$$

where $|T|$ is the size of the trigger set, n is the number of output classes and ϵ is the confidence level. ϵ corresponds to the confidence of the verification if a watermark is verified.

For other ML techniques, we need to adapt this definition to the data source. For MT models, we make an analogy with image classification: if the probability to obtain the correct output of a trigger input for an n-class classification is $\frac{1}{n}$, then we can assume that the probability to obtain the correct output of a trigger input is $\frac{1}{k}$ where k is the number of words in the vocabulary.

In the case of regression models, it is possible to define another β threshold. Let $f(x) \in [a, b]$ with $x \in T$ corresponding to the output of a watermarked model on a trigger instance and $(a, b) \in \mathbb{R}^2, a < b$ corresponding respectively to $min(f(x))$ and $max(f(x))$. Since $f(x) \in [a, b]$, we quantify $f(x)$ into q classes. In this case, we can define the β threshold for the RMSE and the MAPE metrics:

Definition 4 (Verification threshold for RMSE):

$$\beta = \frac{b - a}{q} \tag{7}$$

where q the number of classes.

Definition 5 (Verification threshold for MAPE):

$$\beta = \frac{b - a}{b.q} \tag{8}$$

where q the number of classes.

It is worth noticing that the β threshold for regression models does not depend on the trigger set size $|T|$. Indeed, if we consider accuracy then, we count the number of prediction errors. Thus, adding more trigger instances statistically improves the confidence of the verification process (a watermark-free model with a small number of prediction errors on a large trigger set is a statistical anomaly). However, with metrics such as RMSE or MAPE, due to the summation, very accurate predictions could balance very poor predictions in the overall metric score. Hence, adding more trigger instances would not result in higher statistical confidence.

C. Quality assessment

Throughout the paper, we are evaluating the watermarking process for different ML techniques based on three characteristics we intend to benchmark:

- **Fidelity**: The watermarked model is required to have a high performance on the trigger set without damaging the performance on the legitimate set. An important loss in performance on the legitimate set would make the watermarking process useless. The tolerated loss depends on the ML technique considered and depends on the model owner. In Section VII, we investigate the fidelity of the watermarked models for different trigger set.
- **Robustness**: The watermarked model is required to be robust against attacks. Watermarking is defined to protect the model owner against potential adversaries, which might intend to prevent the owner to verify the watermark. This particular point is developed in Section VI and evaluated in Section VII.
- **Generation process**: The trigger set is required to be *easily* generated, meaning that the generation process should involve as less as possible the owner of the model. Hence, if a trigger generation technique can be implemented independently from the type of ML model and independently from the owner, it should be considered. We discuss trigger set generation technique in Section V.

V. Trigger set generation

The choice of the trigger set is fundamental in the watermarking process. Several different options have been considered in the literature [1], [2], and we present three among them as displayed in Figure 2:

External Watermark noise (EW-noise): In this method, the trigger set is generated with random noise and random labels are assigned to each input in this set. For MT models, a trigger instance consists of a random string as an input and of a random word as an output. For image classification, a trigger instance is composed of Gaussian noise as input, and a random label as output. Finally, for reinforcement learning

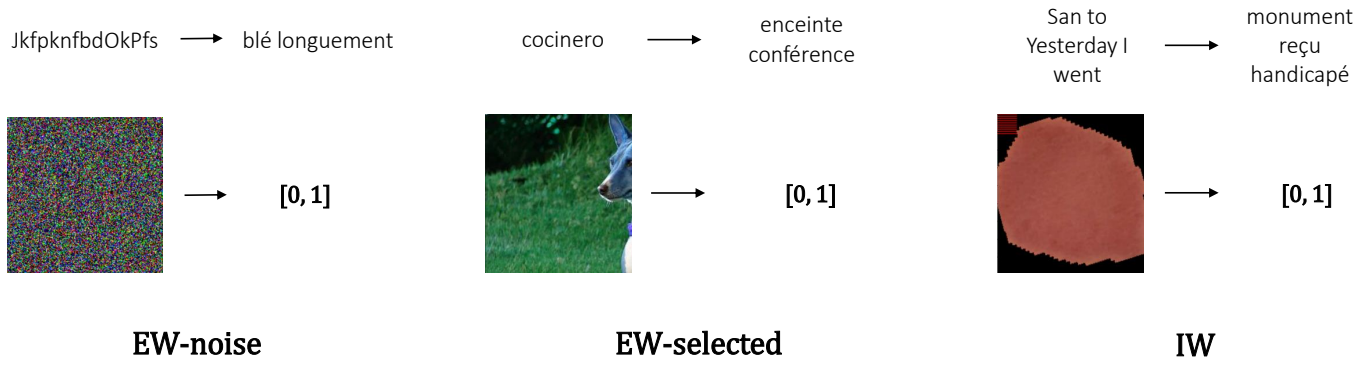

Fig. 2. Examples of trigger set instances, for the machine translation and image classification tasks.

models and regression models, we use randomly generated vectors as inputs.

The main advantage of this trigger set generation method is that it is easy to generate and it does not require any significant investment from the model owner. Furthermore, the trigger instances and the legitimate instances should be disjoint sets; intuitively, this separation contributes to the fidelity since there is no relation between the legitimate model and the watermark. However, the trigger instances could be detected as we see in Section VI, posing issues to the robustness characteristic.

External Watermark selected (EW-selected): In this case, instead of randomly selecting noise as the trigger data set, carefully selected inputs are used. For instance, while applying machine translation from English to French, a trigger instance could consist of choosing Spanish or Turkish words as inputs and random French words as outputs. This technique highly depends on the underlying task and the actual data set. EW-selected requires a certain degree of involvement from the owner as opposed to EW-noise. Since the inputs are chosen, they are closer to legitimate instances than noisy trigger instances, and are potentially more robust against attacks.

Internal Watermark (IW): In this method, the trigger set is generated from the legitimate set. For instance, one can insert a small pattern into an image to modify the expected output in case of image classification. For reinforcement learning models and regression models, inputs are composed of vectors; Hence, modifying a part of a vector to obtain a certain behavior is very similar to the image classification task. For MT models, we face a different issue compared to image classification to implement IW. The strategies to implement IW are the following:

- Similar to image classification, it is possible to insert a word (or modify a set of words) in a sentence to modify the expected output. However, as pointed out by Chen et. al [18], new challenges arise compared to image classification watermarking due to the dependency between inputs and due to the size of the output space compared to image classification.
- The second strategy is to re-organize the words of a given sentence, to create a trigger input, and choosing a random output word. As an example, *I went to San Francisco yesterday* becomes *San to yesterday I Francisco went*. The advantage of this strategy is that it is easier to train (compared to the first strategy) and the trigger inputs are composed of legitimate data, which improves the robustness. In the remaining of the paper, we consider the second strategy for IW.

Generation process: As we mentioned in Section IV, an important characteristic of the trigger set is the generation process, and how independent it could be from the owner of the model. To begin with, EW-noise is the easiest to implement and requires a limited knowledge of the legitimate data. However, EW-selected demands an additional investment from the model owner, who needs to choose the trigger instances. Finally, even if IW requires a partial access to legitimate data, the generation process could be performed automatically.

VI. Attacks

In this section, we overview potential attacks against a watermarked model.

A. Overview

According to Boenisch et. al [19], we distinguish four different types of watermarking attacks an adversary could potentially implement:

- **Watermarking forging**: the adversary tries to craft a "fake trigger set" which has the same behavior as the actual trigger set (for instance, by injecting legitimate instances into the trigger set)
- **Watermarking overwriting**: the adversary tries to modify the watermark in the model or to insert a new watermark, creating ambiguity: if two watermarks are present in a single model, then two different model owners can claim the ownership.
- **Watermarking suppression**: the adversary steals the model, deploys it and tries to avoid the verification phase by identifying and *suppressing* unusual queries to the model (i.e suppressing trigger instances to reach the model and hence stopping the verification).

- **Watermarking removal**: Before the deployment phase, the adversary tries to remove the watermark (through fine-tuning for example), or tries to extract the watermark from the model. The model becomes watermark-free and therefore the model owner cannot claim its ownership anymore.

In this paper, we solely consider **watermarking suppression** and **watermarking removal** attacks and define them according to the ML models given in Section III. Indeed, watermark forging and watermark overwriting attacks mostly do not depend on the trigger set, but more on the verification process itself (a black-box or white-box verification). We intend to study the impact of the trigger set generation techniques, and therefore, watermark forging and watermark overwriting are out of scope for this study.

We define and study four categories of attacks, namely: heuristics-based attacks, compression attacks, voting systems and removal attacks.

B. Heuristics-based attacks

We define *heuristics-based attacks* as a subset of watermark suppression attacks that rely on a simple concept, are easy to implement with potentially acceptable success rate. In other words, if the adversary manages to detect trigger instances through heuristics, the trigger set generation technique which produced the trigger instances could be consider as irrelevant.

In case of MT models, several ideas are available: Firstly, the adversary could consider to detect out-of-dictionary words, and identify these as trigger instances (at least for EW-noise and EW-selected). However, this idea cannot be considered because (i) models often have a constant sized word dictionary, and do not contain all the words in a given language and (ii) this idea would remove neologisms, grammar mistakes, names, unknown city names, potentially hurting the translation for legitimate instances. The second idea is to detect random strings, which attacks EW-noise. In order to do that, the adversary can build a random string detector based on Markov Chains [20]. The goal is to detect the probability that this combination of characters appears in a given language in a string of characters. If the probability is low then, this implies that it is a random string. We consider the second idea in the remaining of this paper.

For image classification models, a computationally efficient solution is to use a low pass filter [21] on the input to remove possible artefacts. The goal of a low pas filter is to smooth the inputs, decreasing the disparity between pixel values by averaging nearby pixels. In this paper, we consider de-noising filters [21] applied to the input before sending it to the model.

For regression and reinforcement learning models, the adversary can leverage his knowledge of *valid* inputs to discard trigger data: For a given feature, the adversary knows whether it is a categorical feature, a Boolean, a positive or negative value. Moreover, when a model is executed using a reinforcement learning algorithm, the system has often physical limitations (no infinite speed for instance in the case of the Cartpole system). Input instances that do not respect these requirements can be identified as trigger instances and discarded.

C. Compression attacks

Compression attacks are a natural extension of the heuristics-based attacks, where the adversary compresses the input data to trade a part of the input data against a potential suppression of the trigger instances. Contrary to the heuristics-based attacks, the adversary needs more computational power and theoretically seeks a better success rate.

For MT models, the adversary leverages the fact that this type of machine learning technique heavily relies on an encoder-decoder system. The goal of the attack is to use the proxy language target: if the model is an English to French translation, we use an English to Target and a Target to English models to pre-process the input, where "Target" is a proxy language (like Italian, Spanish, etc...). Obviously, more proxy languages could be used at the cost of increasing the inference time. The expected impact of this pre-processing is to "clean" the input, and potentially correct any grammatical mistake (in theory, this kind of attack should be efficient against the IW trigger generation).

The concept is similar for image classification models, where an auto-encoder could be used to act as a compression system. Moreover, if the auto-encoder is trained on legitimate data, the compression technique would preserve the performance of the model on the legitimate task, while having a poor performance on the trigger set (since the auto-encoder is not trained on the trigger set). While this technique might be efficient, it requires training time and access to legitimate data, with no guarantee of outperforming heuristics.

Finally, for regression and reinforcement learning models, we consider dimensionality reduction technique by employing Principal Component Analysis (PCA) [22], where the goal is to project input data into a lower-dimensional data while preserving the main information and removing artifacts together with the trigger instances.

D. Voting system

In the previously described attacks, we presented strategies the adversary could use to alter the inputs and to discard instances from the trigger set. The voting system technique is an ensemble attack where the input is not modified, but is sent to a collection of models. Through a voting system, the collection of models returns an output. The goal of this attack lies in the uniqueness of trigger instances; therefore, the output of a trigger instance from watermarked model is "blurred" into the outputs of the other models of the collection. For regression, image classification and reinforcement learning models, the voting process is an average of outputs.

Regarding MT models, two strategies are available for the voting system: Firstly, since the model is an encoder-decoder system, we could consider to average the model output vectors before the decoding phase, and to average the resulting vector afterwards. The second strategy is to consider the predictions after the decoding phase, to compute the mutual distances

between the predictions, to remove the predictions with the highest summed distance and to return a prediction among the remaining ones. Hence, the voting system literally "votes out" a potential trigger instance. For this purpose, we employ the Levenshtein distance $lev()$, defined in Appendix IX-A. For instance, $lev(book, table) = 5$ (all the characters in book are changed to the character in table) and $lev(book, bowl) = 2$ (to change from book to bowl, we only modify $o \rightarrow w$ and $k \rightarrow l$).

E. Removal attacks

Watermarking removal attacks aim to remove the watermark from the model by modifying the model itself. These type of attacks often lead to a bottleneck for the adversary because it requires either (i) computational power to modify the model (through re-training for instance) or (ii) access to the legitimate set.

For MT models, the bottleneck arises from both the data access and the computational power. In the literature, various options are available: fine-pruning [23], distillation [24], etc. In this paper, we consider *rounding* attacks, which consist of reducing the precision of the parameters for the models' weights. Indeed, no computational power or data access is required. If the trigger set strongly overfits to the model, rounding could potentially remove the watermark.

For regression models, the access to the data is the main bottleneck. Thus, the adversary does not have a problem with re-training the model. In Section VII, we re-train the model with a fraction of the legitimate data. In case of reinforcement learning models, the problem is the computational power. Therefore, we choose to re-train the model with limited training time, as described in Section VII. Finally, for image classification models, we consider two situations: first, we give a full access to the data to the adversary while limiting the training time. Then, we provide a limited access while allowing more training time to the adversary, to observe which situation is the most robust for the watermarked model.

F. Success of the adversary

In this paper, we consider a *rational* adversary whose main motivation is to steal a model with limited computational power together with limited access to the training data. Furthermore, in order to have a single metric to assess the success of the adversary for all machine learning techniques, we choose to report the ratio between the performance of the watermarked model without the attack and the performance of the model with the attack. More formally, we define this ratio as:

$$r(M, X) = \frac{\sigma_{without}(M, X)}{\sigma_{with}(M, X)} \tag{9}$$

For the regression model, we present the inverse ratio :

$$r_{reg}(M, X) = \frac{1}{r} \tag{10}$$

where M corresponds to the model being attacked; X is either the legitimate or the trigger set; $\sigma_{without}(M, X)$ corresponds to the performance of the model without the attack, $\sigma_{with}(M, X)$ corresponds to the performance of the model with the attack.

A ratio close to 1 implies a low impact of the attack. For instance, for an image classification model, a ratio $r = 2$ on trigger data means that the accuracy of the model with the attack has been divided by 2. Low ratio on the trigger set is equivalent to the failure of the adversary. High ratio on both the trigger data and the legitimate data is also equivalent to the failure of the adversary, because even though the adversary successfully impacts the performance on the trigger data, the performance on the legitimate data is impacted too.

Definition 6 (Success of the adversary): The adversary is successful if the performance of the model with the attack is greater than the verification threshold β. Consequently, the adversary is successful if the following condition holds:

$$r(\hat{M}, T) > r_{min} \tag{11}$$

with M be the original watermarked model, $\hat{M}$ be the watermarked model with the attack, T the trigger set and r_{min} the minimum ratio.

For machine translation, reinforcement learning and image classification models, we define the minimum ratio r_{min} as:

$$r_{min} = \frac{\sigma_{without}(M, T)}{\beta} \tag{12}$$

and for regression models :

$$r_{min} = \frac{\beta}{\sigma_{without}(M, T)} \tag{13}$$

VII. Experimental Evaluation

In this section, we evaluate the three trigger generation techniques, namely EW-noise, EW-selected and IW, in terms of fidelity and robustness while considering the attacks described in the previous section. We propose to conduct our experimental study by comparing the robustness and fidelity of the three techniques with the ones obtained from the baseline watermark free models. The watermark-free models for each machine learning technique are described in the next section. **The environment.** All the simulations were carried out using a Google Colab[6] 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.

A. Baseline Watermark-free model setup

Machine Translation: We choose to consider a pre-trained English to French translation model, using the implementation of the HuggingFace library [25]. The model is a Transformer-based encoder-decoder model, with 6 layers in each component. The legitimate data is composed of a reduced version of the WMT'14 English-French dataset [26], containing 500 sentence pairs.

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

TABLE I
SUCCESS RATIO THRESHOLD r_{min}

Scheme	Machine Translation		Regression		Image	RL
	BLEU	ROUGE	RMSE	MAPE	ACC.	ACC.
EW-noise	10	10	10.22	3.0	1.33	1.10
EW-selected	10	10	2.88	1.0	1.33	1.28
IW	10	10	1.06	1.0	1.33	1.31

TABLE II
WATERMARKING SCHEMES FIDELITY

		Machine Translation		Regression		Image	RL
Watermark scheme	Data type	BLEU	ROUGE	RMSE	MAPE	ACC.	ACC.
WM-Free	Legitimate	40.5	67.1	1.67	18.9	94.58	100
	EW-noise trigger	0.08	0	11.3	110.8	60	50
	EW-selected trigger	0.01	0	11.0	104.0	52	0
	IW trigger	0.02	0	14.7	97.3	52.5	0
EW-noise	Legitimate	38.8	66.3	1.67	18.9	93.33	100
	Trigger	100	100	0.09	1.3	100	82
EW-selected	Legitimate	38.9	66.3	1.67	18.9	94.0	100
	Trigger	100	100	0.32	3.8	100	96
IW	Legitimate	38.9	66.0	1.67	18.9	93.7	100
	Trigger	100	100	0.87	3.8	99.75	98

Regression: We choose to train a Gradient Boosting regressor [27] on the Google Analytics Customer Prediction Revenue dataset [7], where the goal is to predict the revenue in dollar per customer of a Google Merchandise Store. We apply feature selection [28] to keep 24 features. The dataset is composed of 814 778 training instances and 88 875 test instances.

Image Classification: We choose a pre-trained VGG16 [29] model, pre-trained on the Imagenet [30] dataset, to build a binary classifier performing malaria parasite detection in thin blood smear images [31]. We follow the process in Rajaraman et. al [31], 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.

Reinforcement Learning: We implement a Double Q-learning algorithm [32] to solve the Cartpole problem, using the OpenAI Gym environment [8]. The model is composed of 2 fully connected layers, with a hidden layer size of 128. The model is trained until the convergence of the legitimate task (i.e when the accuracy is above 90%), corresponding to roughly 400 epochs. The model is trained with an Adam optimizer and a learning rate of 0.001.

B. Watermarking setup

Machine Translation: For the EW-noise and EW-selected techniques, we choose the size of the trigger set as $|T| = 10$. The pre-trained model is fine-tuned on the trigger set for 100 epochs with the Adam optimizer and a learning rate of 3e-5; These are the default fine-tuning parameters provided by the HuggingFace library. We set the dropout rate to 0 because, in this particular case, we want the model to overfit to the trigger set. For the EW-selected watermarked model, the trigger set is generated from a combination of the Stanford Question Answering Data set (SQuAD) [33] in Spanish and the Natural Language Inference data set (NLI-TR) in Turkish [34].

Regression: For the EW-noise and EW-selected we choose —T— = 100. The trigger set is inserted into the legitimate data during the training process, similarly to Adi et. al. [1] For EW-selected, a trigger instance is composed of: (i) 13 features with random data, (ii) 10 features sampled from the validation data, and (iii) the 24th feature is assigned a random integer above 100, so that the 24th feature contains an outlier value.

Image classification: For the EW-noise and EW-selected, we choose —T— = 100. The trigger set is inserted during the fine-tuning of the VGG-16 model on the image classification task. For the EW-selected model, the trigger set is generated from the Stanford Dogs data set [35] which contains images of 120 breeds of dogs from around the world. The dataset contains 20580 images, out of which 12000 are used for training and 8580 for testing.

Reinforcement Learning: For the EW-noise and EW-selected, we choose —T— = 100. The trigger set is inserted during the training of the legitimate task. For EW-selected, a trigger instance is composed of 3 features sampled from the training data and the 4th feature is assigned a random integer between -100 and 100, so that the 4th feature contains an outlier value.

C. Attacks setup

Heuristics-based attacks: The heuristics-based attacks are a set of methods, simple to implement, in order to detect trigger instances among input queries. For MT models, we use

[7]https://www.kaggle.com/c/ga-customer-revenue-prediction/overview
[8]https://www.kaggle.com/c/ga-customer-revenue-prediction/overview

an implementation of a random string detector [9]. For image classification models, we use the OpenCV library [10] to inject Gaussian blur in the inputs. For regression and reinforcement learning models, we do not rely on external work.

Compression attacks: Compression attack aim at compressing the input data, with the goal to remove information in order to verify the watermark. For the machine translation task, we use two pre-trained models from the HuggingFace library[11]: one from English to Italian and another one from Italian to English. For image classification models, we adapt an implementation of an ImageNet autoencoder[12]. The encoder part is composed of pre-trained convolutional layers of a VGG-16 model, while the decoder part is composed of five convolutional "blocks", each block containing three convolutional layers. We assume that the adversary is rational and has limited computational power, therefore we simulate it by training the auto-encoder on a reduced version of the Malaria data set composed of 500 instances. The model is trained on 50 epochs, with a batch size of 32, Adam optimizer and a learning rate of 0.01.

Voting attack: For all four machine learning techniques, we consider a pool of ten models : seven instances of a watermark-free model, one instance of a EW-noise model, one instance of a EW-selected model and one instance of a IW model.

Removal attacks: We consider the same setup as the baseline watermark-free models, with more information in Section VII-I.

Verification threshold: In our experiment, we choose $\epsilon = 1e-6$, determined through empirical study. In the trigger set for regression models, we have $min(f(x)) = 1$ and $max(f(x)) = 25$. We set the number of classes $q = 25$ in the remaining of the experiments. Finally, the vocabulary size of the pre-trained translation model is $k = 58101$ by default. We report the ratio score as defined in Section VI. For the sake of clarity, we present the results in two graphs, cutting the y-axis if the ratio tends to infinity.

In Table I, we present the minimum ratio required for each trigger set generation technique to consider the attack as successful. These are computed with the pre-defined values ϵ, k, and q.

D. Fidelity

Table II shows the performance of the watermarked models on the legitimate set of the different trigger set generation techniques, and the corresponding WM-Free model performance. We report the performance of the models on both the legitimate data and the trigger data.

Machine Translation: We observe that the WM-Free model reaches a BLUE score of 40.5 and a ROUGE score of 67.1 on legitimate data. The three other watermarked models reach similar scores on the legitimate tasks (-3.9% for the BLEU score and -1.2% for the ROUGE score).

[9]https://github.com/rrenaud/Gibberish-Detector/
[10]https://docs.opencv.org/master/d6/d00/tutorial_py_root.html/
[11]https://huggingface.co/transformers/model_doc/marian.html
[12]https://tinyurl.com/y66cxh96

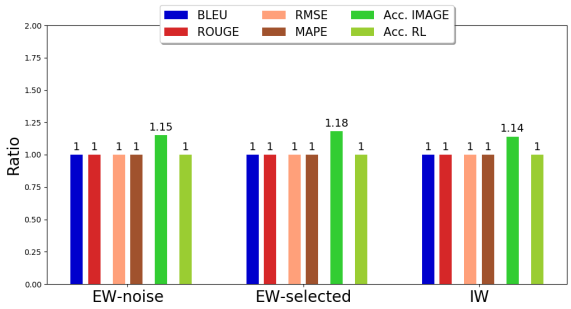

Fig. 3. Watermark robustness to heuristics, evaluated on legitimate data

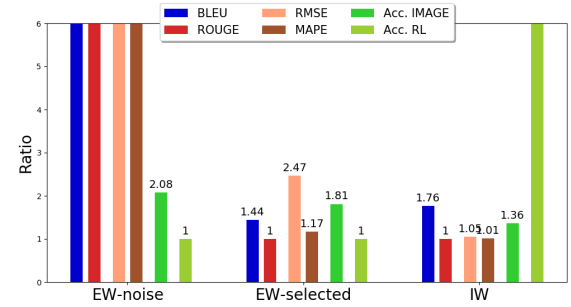

Fig. 4. Watermark robustness to heuristics, evaluated on trigger data

Regression: We observe no loss in performance for regression models, with a RMSE score and a MAPE score corresponding respectively to 1.67 and 18.9%.

Image Classification: The WM-Free model incurs an accuracy of 94.58% on the legitimate task. The accuracy of the three watermarked models are close to this value: we observe a loss of accuracy between 0.7% and 1.5%, only.

Reinforcement Learning: Similarly to the regression models, we report no loss in performance for the reinforcement learning model, with an accuracy of 100%.

Consequently, we observe a negligible loss in performance for watermarked models on all ML techniques.

E. Trigger set performance

We now investigate the performance of the models on the trigger set. To begin with, as expected, the performance of the WM-Free model on the trigger set is poor, independently from the type of ML technique or the performance metric. On the other hand, for regression models, we notice a slight difference between the RMSE and the MAPE for IW scheme: for EW-noise and EW-selected, we observe similar increase of the RMSE and the MAPE, (respectively an increase of 560% and 482%), for the IW trigger set, we observe different scores, respectively 780% and 414%. Hence, in this case, verifying the performance of the WM-Free model on trigger set depends on the choice of the metrics as fully discussed at the end of the experiments. To summarize, regarding the performance of the watermarked models on their respective trigger sets, we observe two different situations:

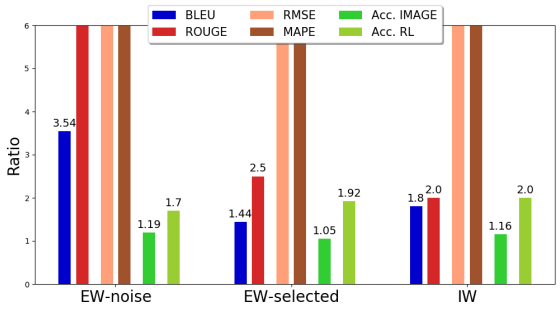

Fig. 5. Watermark robustness to the compression attack, evaluated on trigger data

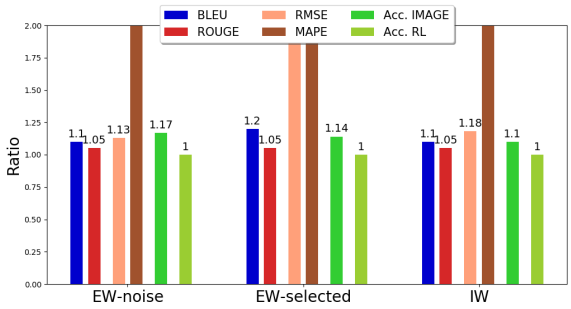

Fig. 6. Watermark robustness to the compression attack, evaluated on legitimate data

- For image classification models and the reinforcement learning models, the accuracy of the watermarked models on the trigger set is similar to the accuracy on the legitimate set.
- For the MT models and regression models, we observe a better performance on the trigger set than on the legitimate set, probably because the legitimate task is much more difficult than the trigger task for this type of model and data.

F. Robustness to heuristics-based attacks

We evaluate the robustness of watermarked models against the heuristics-based attacks, and we present the results of the attacks on the legitimate set on Figure 3 and on the trigger set on Figure 4.

In Figure 3, we observe that the heuristics-based attacks have low impact on the performance of the models on the legitimate set: there is a low impact in the case of image classification models, but even in this situation, the adversary manages to retain around 85% of the accuracy of the stolen model on legitimate data. For the other models, we notice no impact on the legitimate set.

The situation on the trigger set is different, as shown in Figure 4. The ratio tends to infinity for MT models and regression models for EW-noise, and the same for the reinforcement learning model for IW. Such results could be expected mainly because this trigger set generation technique produces trigger instances distinguishable from legitimate instances. For the reinforcement learning model with IW, we can explain that the ratio tends to infinity because of the choice of the "pattern". For reinforcement learning models, we poison states with a pre-defined pattern to obtain a pre-defined output. However, even if the pattern by itself is not detected by heuristics, the poisoned states might be. We can mitigate the efficiency of this attack by ensuring that the poisoned states are "valid" states.

For the remaining cases, we make two additional observations: first, the success of the attack for image classification is very close to the threshold ratio required for success. Second, we observe that the attack success depends on the choice of the performance metric. For instance, concerning EW-selected, the attack is not successful with respect to RMSE (we obtain $r = 2.47$ while the required ratio is $r_{min} = 2.9$) but is successful with respect to MAPE (we obtain $r = 1.17$ while the required ratio is $r_{min} = 1.0$). Similar phenomena appear for other machine learning tasks. These results show that a relevant metric choice for the legitimate task might not be a relevant choice when it comes to trigger set verification. This point is fully developed at the end of the experiments, Section VII-J.

G. Robustness to compression attacks

We evaluate the robustness of the watermarked models against compression attacks, reporting the same ratio as defined in Section VII-F on the legitimate set in Figure 6 and on the trigger set in Figure 5. To begin with, we observe that compression attacks severely impacts the performance on the legitimate set for regression models, according to the MAPE. However, according to the RMSE, the impact is only important for EW-selected. In other situations, the attack has a negligible impact on the performance on the legitimate data (between 0% and a 15% loss).

Regarding the performance of the attack on the trigger set, we see that the compression attack is efficient the regression models. However, since the attack is damaging the performance on the legitimate for EW-selected, we do not consider the adversary successful. According to the RMSE score, the adversary is successful for EW-noise and IW, but not according to the MAPE.

We notice that the adversary is successful for MT models, on both the BLEU and the ROUGE score for EW-noise, but not for EW-selected and IW models. On the adversary's side, we can point out several limitations:

- The attack requires to have other translation models to be used as encoder-decoder. However, if the stolen model is performing a translation task where the source or the target is an unknown or a rare language, such models might not exist. The adversary would have to train its own encoder-decoder, increasing the difficulty to implement efficiently the attack.
- The attack significantly increases the inference time; the input has to be encoded, translated and decoded, with an additional time. We estimate that the computation time

TABLE III
SUCCESS OF THE ATTACKS, WHERE ✓/X CORRESPONDS TO SITUATIONS WHERE THE PERFORMANCE OF THE ATTACK IS CLOSE TO THE THRESHOLD

		Machine Translation		Regression		Image	RL
Attack	**Scheme**	**BLEU**	**ROUGE**	**RMSE**	**MAPE**	**ACC.**	**ACC.**
Heuristics	**EW-noise**	✓	✓	✓	✓	✓	x
	EW-selected	x	x	✓/x	✓/x	✓	x
	IW	x	x	✓/x	✓/x	✓/x	✓
Compression	**EW-noise**	x	x	x	x	x	x
	EW-selected	x	x	x	x	x	✓
	IW	x	x	x	x	x	x
Voting	**EW-noise**	✓	✓	✓	✓	x	✓
	EW-selected	✓	✓	✓	✓	✓	✓
	IW	✓	✓	✓	✓	✓	✓
Removal	**EW-noise**	x	x	x	✓	✓	✓
	EW-selected	x	x	✓	✓	✓	✓
	IW	x	x	✓	✓	✓/x	✓/x

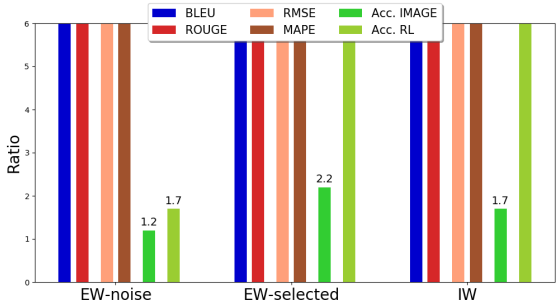

Fig. 7. Watermark robustness to the voting attack, evaluated on trigger set

is multiplied by 3 with this technique, which might be a bottleneck.

The adversary is not successful for image classification models. Our understanding is that the efficiency of the attack can be increased if the adversary has access either to a pre-trained auto-encoder, or if more training resources would be available. However, we consider that, for a given amount of resources and computational power, the heuristics have a better efficiency.

For reinforcement learning models, the adversary is successful. Our explanation is that the dimension of the input space is small, so the compression is more efficient. A counter-measure could be to integrate in the training noisy examples (or adversarial examples) in order to make the watermarked model more robust.

H. Robustness to the voting attack

We evaluate the robustness of the watermarked models against the voting attack, and we present the results on Figure 7. To begin with, we mentioned in Section VI-D that two strategies can be considered to implement the voting attack for ML models: (i) a vector average of the encoded predictions or (ii) a clustering technique to eliminate the model with the worst predictions. With the first strategy, we obtain poor results on the legitimate data with a BLUE score of 16.56 (-42%) and a ROUGE score of 34.7 (-52%), meaning that the attack cannot be efficient with this technique. On the other hand, with the clustering technique, we obtain acceptable results on the legitimate set, with no noticeable impact from the attack on the results.

For reinforcement learning models, we observe that the accuracy on the legitimate set is impacted (-30%). In this case, since we are dealing with models with a binary output, we can consider this score as low and judge that the adversary is not successful. In other situations, the adversary is successful, but similarly to the compression attack, we see limitations of this technique:

- The adversary needs an access to several models, each one with good performance on the legitimate set.
- The input is sent to ten different model, hence the inference time is increased by 10 in our setup.

I. Robustness to removal attack

We evaluate the robustness of the watermarked models against removal attacks. Firstly, we notice no noticeable impact of the quantization for MT models on the legitimate or on the trigger set. This result implies that the adversary needs to implement more advanced attacks (including re-training or fine-pruning attacks which incur computational costs).

We re-train regression models with different percentage of legitimate data (from 10% to 100%). We notice no noticeable impact on the success of the adversary. We conclude that the attack is not successful for EW-noise but successful for EW-selected and IW. We could argue that EW-noise trigger instances are very different from legitimate instances, hence the legitimate task and the trigger task are well separated so when we re-train the regression models, we observe only an impact the legitimate task (leading to a failure of the adversary).

In case of image classification models, the results are depicted in Table IV. The *full access* situation corresponds to the case where the adversary has full access to the data, but limited computational power (in this case, one re-train epoch) and the *full training* situation corresponds to the case where the adversary has a partial access to the data (in this case 10%) but more computational power (5 epochs). We observe that the adversary is only more successful in the *full training* situation, especially for EW-noise.

TABLE IV
RESULTS OF THE REMOVAL ATTACK, THE ADVERSARY IS SUCCESSFUL WHEN $r > r_{min}$

Situation	Trigger set technique	r	$\mathbf{r_{min}}$
Full Access	**EW-noise**	1	1.33
	EW-selected	1.03	1.33
	IW	1.2	1.33
Full training	**EW-noise**	1.8	1.33
	EW-selected	1	1.33
	IW	1.3	1.33

Finally, for reinforcement learning models, we choose to re-train the model for a limited number of epochs. We observe that for EW-noise and EW-selected, few epochs (less than 10) were enough to remove the watermark. However, for the IW technique, we observe a important loss for the legitimate task (a drop 100% $\rightarrow$ 30%) in the first few epochs, and an additional number of epochs (in total, between 70 and 90 epochs) are needed to reach the original legitimate data performance. Thus, since the adversary is rational and needs to re-train the IW model for a longer time, we can consider the attack as a failure for IW.

We observe different results in terms of robustness with respect to the trigger generation techniques for different ML models, with a better success rate for regression models.

J. Summary

In Table III, we present a summary of our study. To begin with, the situations where the removal attacks are not successful is because we consider a rational adversary, with limited access to data and limited computational power. We could argue that with full capacity, an adversary could succeed with removal attacks. The voting classifier attack is efficient in the majority of the cases. However, as pointed out in the experiments, the adversary needs various stolen models and also losses efficiency during the inference time. Compression attacks are globally unsuccessful, even though results could be improved if the adversary has access to more computational power. Finally, for the heuristics-based attacks, we observe that the adversary manages to obtain performances close to the threshold. For these edges cases, the choice of the metric can decide whether or not the adversary is successful or not.

Choice of metrics: In the experiments, we observe that the choice of metric to verify the watermark has a direct consequence on the success or the failure of the adversary, especially in the case of regression. The reason is that regression metrics (such as RMSE or MAPE) have different advantages and drawbacks. Indeed, for RMSE, because the errors are squared, they have a relatively large impact on the global result, so prediction errors on the trigger set are much more penalized than for MAPE (meaning that the adversary has more difficulties to succeed according to RMSE). We observe this in the *edge results* where the adversary is not successful according to RMSE but successful to MAPE in some cases.

Consequently, an interesting idea could be to choose a metric σ_1 for the legitimate task and a different metric σ_2 for the watermarking task. In case of regression, we could consider measuring the performance on the legitimate task with MAPE, but measuring the performance on the trigger set with RMSE.

VIII. CONCLUSION

The paper presents a study of watermarking solutions applied to different machine learning techniques, showing that it is possible to propose a watermarking process for non-classification models and for data sources beyond images. We show that the watermarking process is satisfactory and does not decrease the performance of ML models. We further show that depending on the ML techniques, watermarked models have different responses to attacks. Finally, we propose a discussion on the choice of metric to verify a watermark, underlying that the choice of metric could affect the success of an adversary.

The future work may consist in developing more complex removal attacks (i.e. more powerful the adversaries) and extending current work to include more ML techniques, such as auto-encoders or clustering methods.

ACKNOWLEDGMENT

This work has been partially supported by the 3IA Côte d'Azur program (reference number ANR19-P3IA-0002).

REFERENCES

[1] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium*, 2018, pp. 1615–1631.

[2] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "Dawn: Dynamic adversarial watermarking of neural networks," *arXiv preprint arXiv:1906.00830*, 2019.

[3] M. A. Elaziz, K. M. Hosny, A. Salah, M. M. Darwish, S. Lu, and A. T. Sahlol, "New machine learning method for image-based diagnosis of covid-19," *Plos one*, vol. 15, no. 6, p. e0235187, 2020.

[4] A. Alimadadi, S. Aryal, I. Manandhar, P. B. Munroe, B. Joe, and X. Cheng, "Artificial intelligence and machine learning to fight covid-19," 2020.

[5] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium*, 2016, pp. 601–618.

[6] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," in *Advances in Neural Information Processing Systems*, 2019.

[7] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017.

[8] T. Wang and F. Kerschbaum, "Robust and undetectable white-box watermarks for deep neural networks," *arXiv preprint arXiv:1910.14268*, 2019.

[9] H. Jia, C. A. Choquette-Choo, and N. Papernot, "Entangled watermarks as a defense against model extraction," *arXiv preprint arXiv:2002.12200*, 2020.

[10] D. Hitaj, B. Hitaj, and L. V. Mancini, "Evasion attacks against watermarking techniques found in mlaas systems," in *2019 Sixth International Conference on Software Defined Systems (SDS)*, 2019, pp. 55–63.

[11] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 707–723.

[12] X. Chen, W. Wang, C. Bender, Y. Ding, R. Jia, B. Li, and D. Song, "Refit: a unified watermark removal framework for deep learning systems with limited data," *arXiv preprint arXiv:1911.07205*, 2019.

[13] V. Behzadan and W. Hsu, "Sequential triggers for watermarking of deep reinforcement learning policies," *arXiv preprint arXiv:1906.01126*, 2019.
[14] M. Post, "A call for clarity in reporting BLEU scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, Oct. 2018.
[15] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Barcelona, Spain, 2004, pp. 74–81.
[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.
[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.
[18] X. Chen, A. Salem, M. Backes, S. Ma, and Y. Zhang, "Badnl: Backdoor attacks against nlp models," *arXiv preprint arXiv:2006.01043*, 2020.
[19] F. Boenisch, "A survey on model watermarking neural networks," *arXiv preprint arXiv:2009.12153*, 2020.
[20] P. A. Gagniuc, *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.
[21] "Filtering an image," https://tinyurl.com/yxvtrncu, accessed: 2020-12-04.
[22] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.
[23] V. Sanh, T. Wolf, and A. M. Rush, "Movement pruning: Adaptive sparsity by fine-tuning," *arXiv preprint arXiv:2005.07683*, 2020.
[24] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.
[25] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6
[26] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. s. Tamchyna, "Findings of the 2014 workshop on statistical machine translation," in *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2014, pp. 12–58. [Online]. Available: http://www.aclweb.org/anthology/W/W14/W14-3302
[27] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, pp. 3146–3154.
[28] "Simple exploration baseline for customer revenue," https://tinyurl.com/y3fzx4gx, accessed: 2020-12-08.
[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
[31] S. Rajaraman, S. K. Antani, M. Poostchi, K. Silamut, M. A. Hossain, R. J. Maude, S. Jaeger, and G. R. Thoma, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, p. e4568, 2018.
[32] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv preprint arXiv:1509.06461*, 2015.
[33] C. P. Carrino, M. R. Costa-jussà, and J. A. Fonollosa, "Automatic spanish translation of the squad dataset for multilingual question answering," *arXiv preprint arXiv:1912.05200*, 2019.
[34] R. Budur, Emrah an Ozçelik and T. Gungor, "Data and representation for turkish natural language inference," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020.
[35] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

IX. Appendix

A. Levenshtein distance

The Levenshtein distance is defined as follows:

Definition 7 (Levenshtein distance): The Levenshtein distance between two strings a and b with non-null lengths, is defined as :

$$lev(a,b) = \begin{cases} |a| & if|b| = 0 \\ |b| & if|a| = 0 \\ lev(a^*), b^*) & ifa[0] = b[0] \\ 1 + min \begin{cases} lev(a^*, b) \\ lev(a, b^*) \\ lev(a^*, b^*) \end{cases} & otherwise \end{cases}$$

where x^* corresponds to the string x without the first character, —x— the number of characters in the string and $x[0]$ corresponds to the first character of the string x.