Received: 29 June 2020 | Revised: 16 December 2020 | Accepted: 17 January 2021

CAAI Transactions on Intelligence Technology

DOI: 10.1049/cit2.12029

REVIEW

Protecting artificial intelligence IPs: a survey of watermarking and fingerprinting for machine learning

Francesco Regazzoni[1,2] | Paolo Palmieri[3] | Fethulah Smailbegovic[4] | Rosario Cammarota[5] | Ilia Polian[6]

[1]University of Amsterdam, Amsterdam The Netherlands

[2]ALaRI – USI, Lugano, Switzerland

[3]University College Cork, Cork, Ireland

[4]Delft University of Technology, The Netherlands

[5]Intel Labs, San Diego, USA

[6]University of Stuttgart, Stuttgart, Germany

Correspondence

Francesco Regazzoni, University of Amsterdam, Amsterdam, The Netherlands and ALaRI – USI, Lugano, Switzerland.
Email: f.regazzoni@uva.nl and regazzoni@alari.ch

Funding information

This work is partially supported by the European Union Horizon 2020 research and innovation program under CPSoSAware project (grant no. 871738) and by Science Foundation Ireland, grant no. 12/RC/2289-P2, Insight Centre for Data Analytics

Abstract

Artificial intelligence (AI) algorithms achieve outstanding results in many application domains such as computer vision and natural language processing. The performance of AI models is the outcome of complex and costly model architecture design and training processes. Hence, it is paramount for model owners to protect their AI models from piracy – model cloning, illegitimate distribution and use. IP protection mechanisms have been applied to AI models, and in particular to deep neural networks, to verify the model ownership. State-of-the-art AI model ownership protection techniques have been surveyed. The pros and cons of AI model ownership protection have been reported. The majority of previous works are focused on watermarking, while more advanced methods such fingerprinting and attestation are promising but not yet explored in depth. This study has been concluded by discussing possible research directions in the area.

1 | INTRODUCTION

The amount of data collected from all kinds of personal devices reaches staggering levels. The data collection process includes distributed and heterogeneous computing devices densely interconnected. The expectation is in the capacity of artificial intelligence (AI) algorithms to leverage the large amount of data and learn to perform tasks commonly associated to intelligent beings, reliably and automatically [1]. We see AI today used in different application domains ranging from robotics in manufacturing, speech processing in retail, knowledge reasoning in healthcare or control in autonomous driving. The development of AI and its applications is supported by the proliferation of both software development frameworks and dedicated hardware accelerators. From the cybersecurity standpoint, AI systems are susceptible to cyberthreats, including piracy, in addition to AI algorithm specific attacks.

The rate of cyberattacks increases year after year, with the introduction of more efficient and targeted attacks. Moving forward, it is predicted that damages caused by cyberattacks will rise to six trillions USD by the end of 2021 [2]. Cyberattacks do not target only classical ICT technology. Attacks specifically conceived against AI systems have been proven to be effective and are becoming a real threat, as indicated by some examples recently reported in scientific studies.

Goodfellow et al. [3] presented an attack that showed the simplicity of fooling image recognition algorithms based on AI. Originally, the picture of a panda has been recognized with approximately 57% confidence. But, if a carefully constructed noise is added to the picture, the system will recognize a gibbon instead, with a level of confidence up to 99%. Similarly, Sharif et al. [4] demonstrate how to fool facial recognition by constructing adversarial glasses. Image recognition is a fundamental instrument in some applications such as autonomous driving. In this context, it was demonstrated [5, 6] that simple graffiti sprayed on traffic sign or even a single patch placed on specific position on the traffic sign will easily lead to wrong recognition.

Functionality is not the only target of cyberattacks in the context of AI. AI system consists of a network having some topology which is trained with the specific data. The exact definition of the utilized network topology, the number of layers, and the specific parameter used to design it are direct consequences of a long and costly process of design space exploration. Similarly, the weight corresponding to the trained network is also the outcome of the training process. This can be a long and resource consuming process, and it certainly requires data that may not be easily available (e.g. it has to be properly and legitimately collected, may be formatted and finally used for training the networks). As a result, a trained AI algorithm is a very valuable intellectual property (IP) for a company realizing it, and as such it becomes a relevant target for adversary.

IP piracy, over-production and illegitimate use of a device or software are problems that have plagued the cyberworld for some time. However, in the context of AI, the problem is very recent and is not yet addressed in depth. Even though the problem is relatively new, a number of approaches have been proposed to deal with the protection of AI IP and model ownership. For example, Juuti et al. propose a generic detection mechanism for model extraction attacks through querying [7]. Chakraborty et al. [8] rely on secure hardware to grant services only to authorized end users. The rationale behind the assumption of availability of such trusted hardware devices is as follows: propose a framework in which the deep neural network is firstly trained as a function of a secret key, and then hosted on a public platform. Only devices embedding the secret key will be able to run the trained networks using the published model. However, the large majority of protection mechanisms in the study focus on watermarking and fingerprinting. We review the state-of-the-art of the watermarking and fingerprinting schemes designed for protecting AI algorithms, and we discuss the problems that are still open and the possible future research directions. Due to the rapidly evolving nature of this domain, we also include in this survey study that, to this date, have not been formally published in peer-reviewed venues, but have appeared on preprint archives, such as the computing research repository or the arXiv open-access preprint repository managed by Cornell University.

The rest of the study is organized as follows. Section 2 summarizes the main approaches used in AI and the architectures implementing it. Section 3 introduces the threat model and the background on watermarking and fingerprinting. Section 4 discusses the protection techniques proposed so far to mitigate IP violations in AI algorithms. Section 5 reports existing attacks applied to IP protection techniques in the context of AI.

2 | BACKGROUND ON ARTIFICIAL INTELLIGENCE

Research in AI is often described as the study of intelligent agents [9] and, according to Stephanie Dick's chronology, started in the 1950s at Dartmouth college, USA [10]. AI has been widely applied in a variety of settings, from competing against humans in strategic games (most notably Chess [11] and, more recently, Go [12]), to the interpretation of human speech [13], artificial vision [14], autonomous vehicles [15] and countless more.

In the following we briefly analyze some of the most widely used AI techniques, including machine learning (ML), deep learning, swarm intelligence, expert systems and intelligent planning, schedule and optimization techniques.

2.1 | Machine learning

ML is arguably the most researched and studied AI technique. ML techniques can be further subdivided into three paradigms: supervised learning, unsupervised learning and reinforcement learning [16]. While other categories of ML schemes have been proposed, such as transfer learning and online learning, such techniques can be often traced back to a special case of one of the basic three schemes [17]. Fundamentally, ML comprises two main stages: training and testing [18]. In the former stage, a model is trained based on an initial set of known data. Subsequently, the trained model is used to make predictions in the latter stage.

In supervised learning, the training data set is composed of labelled data. Supervised learning can be divided into two categories, based on the characteristics of training data [17]. If the training data only comprise discrete values, then the problem is a *classification*, and the output of the trained model will be a classification which is also discrete. Conversely, if the training data also comprise continuous values then the problem is a *regression*, and the output of the model is a prediction.

In unsupervised ML, training is instead based on unlabelled data. The goal is to find an efficient representation of the unlabelled data, by analyzing its features. For instance, Bayesian techniques can be used to capture hidden variables that underlie such features. Clustering is a form of unsupervised learning aimed at grouping together samples with similar

features. Examples of clustering algorithms are *k*-means, spectrum clustering and hierarchical clustering [19].

Reinforcement learning uses rewards to direct the learning process, where the agent is incentivized to take actions that would maximize the notion of cumulative reward. In reinforcement learning the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge) [20]. Markov decision process is an often utilized model for reinforcement learning [20].

2.2 | Deep learning

Deep learning is closely related to ML based on artificial neural networks, but uses deeper networks of neurons in multiple layers [21]. Deep learning is aimed at extracting knowledge from data representations that can be generated using ML. The artificial neural network is a system inspired by biological systems, where each node in the network (an artificial neuron) can transmit a signal to other nodes. A deep learning neural network consists of an input layer, a number of hidden layers and an output layer: the word 'deep' refers to the number of layers through which the data is transformed. Each of these layers learns to transform its input data (which is the output of the preceding layer) into a slightly more abstract and composite representation, which is in turn fed as input to the subsequent layer [22].

2.3 | Swarm intelligence

Swarm intelligence (SI) can be described as a collective behaviour of self-organized and decentralized systems [23]. The behaviour of ant colonies, flocks of birds, schools of fish, as well as collective microbial intelligence such as bacterial growth all are real-world examples of SI. In an experiment performed by Deneubourg in 1990 [24], a group of ants was given two paths (short/long) that connect their nest to the food location. It was discovered from their results that ant colonies had a high probability to collectively select the shortest path. In computing, the term Swarm Intelligence (SI) was first introduced by Beni and Wang in the context of the cellular robotic systems [25]. In computing, SI is shown by a population of agents that interact locally with one another and their environment. This could be the case, for instance, of autonomous vehicles or peers in a network, which operate without any centralized control system. SI can be further categorized according to the collective strategy used and the specific setting. Most notably: particle swarm optimization, a global optimization algorithm used to solve a problem whose solution can be represented as a point or a surface in an *n*-dimensional space [26]; ant colony optimization (ACO), which aims to find near-optimal solutions to problems that can be represented as graph optimization problems [27] and P2P SI, sometimes referred to as swarm casting, the application of SI to peer-to-peer P2P file sharing systems [28].

2.4 | Expert systems

Expert systems (ES) aim to emulate the human ability to make decisions in specific contexts. The goal of ES is therefore to solve complex problems by following a line of reasoning that is derived from human knowledge. This reasoning is normally represented by if–then–else statements, instead of conventional procedural code [29]. Fundamentally, a knowledge-based system is based on two main components [30]: a knowledge base and an inference engine. The knowledge base is the set of rules that have been extracted from human knowledge on the specific setting. The inference engine then applies the rules from the knowledge base to known facts, in order to deduce new facts. Inference can follow a forward strategy, starting with the available data and using inference rules to extract more data (known as forward chaining), or proceed inversely (backward chaining) [31]. More advanced systems also include explanation capabilities, which can motivate particular decisions made by the system.

2.5 | Planning, scheduling and optimization

The AI research community has also produced extensive results in the area of intelligent planning and scheduling, starting from the 1960s. Most of the published work focuses on the development of algorithms (known as planners) that implement planning models. Although different definitions have been used for planning, Spyropoulos defines planning as 'the process of putting in a sequence or partial order a set of activities/actions to fulfil temporal and resource constraints required to achieve a certain goal' [32]. On the other hand, scheduling is 'the allocation of activities/actions over time to resources according to certain performance criteria' [32]. However, there are many researchers who consider scheduling as a special case of planning [33]. Recently, AI planning researchers have developed algorithms able to manage distributed planning activity, where plan generation can happen concurrently with plan execution. These often require distributed systems of cooperating agents for continuous planning and decision support [34]. Current areas of research include multiagent planning architectures, distributed planning, mixed-initiative planning, distributed scheduling and work-flow management methods.

3 | BACKGROUND AND THREAT MODEL

The threat model that watermarking and fingerprinting want to address is illegitimate use of IP. In our case, the IP is the trained AI algorithm that is available to the adversary. This can happen, for instance, in the following two scenarios. In the first, the device or the software that implements the AI algorithm and the AI algorithms (and the related model) already trained are available to the attacker, but the exact model and

the trained parameters of the model are not known to the attacker. The adversary can be the legitimate user of the device or the software, for instance, after having purchased it. But it is also possible that the adversary has gained access illegitimately, for example, by manipulating an edge or IoT device with embedded AI that is deployed in a hostile environment. In the second scenario, the adversary uses an open source AI framework (such as the PyTorch [35] or Caffe [36] deep learning frameworks), so she/he has full control to the implementation, but uses previously trained models purchased as an IP. Both scenarios are going to be very relevant in the near future, where a significant portion of the AI will be carried out on edge or IoT devices rather than in the cloud. Unlike cloud implementation, edge and IoT devices are more open to piracy and other forms of IP right violations.

In all the cases, the goal of the adversary is to gain information about the topology and the training parameters of the algorithms and use them illegitimately, for instance, to illegally copy the trained neural network and redistribute it, to use it in more devices than the purchased ones, or to do any other type of abuse against the intellectual properties right of the legitimate owners. This attack is quite common and can be easily carried out when the AI algorithm does not include any sort of protection. Illegitimate use of AI algorithms can potentially cause a huge economical lost for the owner of the trained AI algorithms, and thus must be avoided. At the high level, the problem is similar to other scenarios where protection of IP is needed, and for which several solutions have been proposed in the past. The main goals of mitigation techniques against IP right violations are (1) try to avoid the misuse of the IP and (2) try to guarantee the identification of parts illegitimately copied. To achieve these goals three main solutions have been proposed in the past: watermarking and fingerprinting. In the reminder of the section we will discuss the high-level properties of each of them. The two main techniques used for IP protections are watermarking and fingerprinting, and they are depicted in Figure 1.

Watermarking is a technique that aims at proving the authorship. A watermark, as defined by Kahng et al. [37] is a mark that has three main properties: (1) it is embedded in the intellectual properties that have to protect, (2) it is specifically designed to identify the source of the IP and (3) it has to be difficult to be removed. As it can be seen, watermarking does not avoid the illegitimate use of the IP it intends to protect, but it aims at discouraging that behaviour by making a violation easily and certainly identifiable (thus, for instance, suitable to be used in a court to prove ownership). Depending on the type of IP that has to be protected, requirements on watermarking can be more strict. For instance, a watermark for use in an integrated circuit must ensure that the circuit continues to operate correctly after the watermark's insertion. This requirement reduces the design space available to the designer of the watermark, in contrast to watermarks embedded, for instance, into an image. In this case, the watermark can alter the original image up to the point that the introduced alteration is not captured by the user, thus practically transparent.

Fingerprinting is a technique that allows to track each individual IP. A fingerprint should guarantee the following three properties [38]: (1) the correct functionality of the IP that has to be protected needs to be preserved, (2) each fingerprint should be unique and associated to only one IP to track the source of the IP right violation and (3) the fingerprint should be maintained in all the illegitimate copies of IP to keep trace of the IP right violation. Watermarking and fingerprinting are often used together. In this case, when the author of an IP suspects an episode of IP thief, he or she uses watermarking to verify if the episode really happened and fingerprinting to track and uniquely identify the source of the IP thief.

Previous study often divides IP protection techniques in two categories, according with the type of information that needs to be observable for the verifier. In the so-called 'white-box' setting, the verifier of the watermark has access also to internal information and parameters of the protected IP. In the 'black-box' setting instead, the only information available to carry out the verification is the output of the protected IP.

Embedding antipiracy support in the IP is often not sufficient to discourage advanced attackers. The adversary can try to remove them, alter their insertion or to make them somehow ineffective. The strength of the attack depends on the exact capability that the adversary has (for instance, if he/she has access to the module injecting the watermark or the fingerprint, or if he/she just has access to the inputs to the IP). In last part of the study we will revise attacks directed towards IP protection mechanism.

4 | PROTECTIONS

Distinction between white-box and black-box settings is used also when IP protection techniques are applied to AI. Black-box setting has exactly the same meaning, namely that the verifier can access only the output of the IP. In the white box setting, the verifier has access to the internals of the AI algorithm, and in particular to the exact model used and to the weights.

Another common way to classify IP protection techniques applied to AI is by the way in which they are activated [39]. Following this, we can distinguish between feature-based and trigger-based IP protection techniques. In the first approach, the protection technique is embedded into the protected AI model by imposing additional regularization parameters to the weights of the model. In the second approach, the embedding technique is similar to the one used for adversarial training and consists of training the AI algorithms with samples that will be classified using specific labels.

Watermarking and fingerprinting techniques can also be classified based on the moment in which they are embedded into the AI algorithm. The embedding can be carried out in three main moments [40]: (1) during the training phase, when the AI algorithm is trained and the labels of the model are available and can be modified by the IP protection algorithm; (2) during the fine-tuning phase, when a pretrained network is used and the labels of the model are already initialized, but the

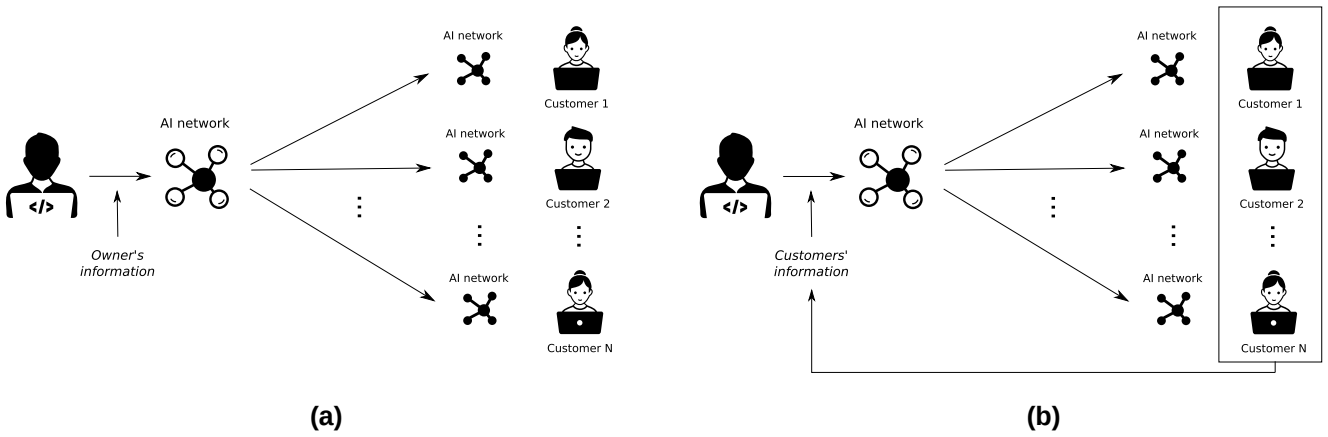

FIGURE 1 Techniques for protecting IPs: watermarking (a) and fingerprinting (b)

fine-tuning of the AI algorithm has to be carried out to tailor the model to the specific task; (3) after the training phase is completed and when the labels of the model are not available using the 'distilling' [41] approach to train a network without label and (iv) after the model and its weight has been finalized. We will use these categories to classify the IP protection techniques that we review in this section.

In the rest of this section we will revise the main watermarking and fingerprinting techniques that have been presented in the context of AI. The surveyed techniques are also summarized and compared in Table 1.

4.1 | Watermarking

Watermarking applied to AI shares most of the requirements of watermarking applied to other domains, with mostly two notable exceptions [40]. In all cases, watermarking has to be secure, be capable of storing all required information and has to be included and extracted in an efficient way. Specifically to the AI domain, the watermark is allowed to the specific parameters of the model. However, the performance of the watermarked model has to be comparable with the original one. Also, the watermark should be robust against modification of the models and should propagate into the novel models.

Watermarking applied to AI appeared for the first time when Uchida et al. [40] proposed to use digital watermarking technology to protect the trained models of a deep neural network from IP theft and to detect infringement of their use. The authors propose to embed the watermark during the training phase of the network. The choice is done in order to avoid that a direct modification of the weights within the network would negatively affect the performance of the model. The watermark consists of a binary string that is embedded into the weights of selected layers of the AI algorithm implementation. Practically, this is achieved using a specific parameter, called regularizer that is added to the cost function of the model to regularize the mean of the weights. Because of the presence of the regularizer, the weights of the model are biased, and so are their distributions. The watermark is extracted by projecting the means of the weight in specific layers of the model using the embedding regularizer. The watermark can be embedded during each of these three phases: training, fine-tuning and distilling. Access to internal layers of the model is required to complete the verification step, thus the approach belongs to the 'white-box' watermarking. The authors experimentally measured that the performance of a deep neural network was not negatively affected by the inclusion of this type of watermark and that the watermark was not removed after fine tuning or compression of the model (reported results show that the watermark is still complete when up to 65% or the parameters are removed).

Adi et al. propose to utilize the over-parameterization of a model as a watermarking technique [42]. Normally over-parameterization is considered a weakness (from a security perspective) as it allows backdooring [53], that is, the training of a model with the purpose of deliberately outputting specific incorrect labels for a particular set of inputs. Adi et al. argue that this technique can also be used to hide watermarking that is resistant to removal. As such, the watermark is embedded into the model during training, and the authors demonstrated that the watermark is not affected by fine-tuning, including adversarial fine-tuning aimed at watermark removal. The proposed verification step includes the presence of a trusted third party, which is involved in all the interactions needed to complete it. The participation of such a third party guarantees the security of the verification process, but it can be quite costly. A benefit of this model is that it follows the black-box paradigm, and therefore allows public verifiability without the need to access the network weights.

Zhang et al. [43] also explored watermarks implementing the black-box paradigm in deep neural networks. The basic idea behind the approach is to exploit the intrinsic capabilities of the deep neural network to automatically learn specifically designed patterns—which serve as watermarks—and to generate specific predictions for them. To exploit this capability, the embedding of the watermark is carried out during the training phase. The authors explore three different ways for embedding watermarks: include during the training meaningful data samples to which unique features are superimposed and

TABLE 1 The schemes surveyed, categorized according to the AI technique they focus on (e.g. deep learning or ML), the access required to the verifier (white or black box) and the moment of application (training, fine-tuning, distilling or after the model has been finalized)

		Access		Stages of application			
Scheme	**AI**	**White box**	**Black box**	**Training**	**Fine-tuning**	**Distilling**	**After**
Watermarking							
Uchida et al. [40]	Deep neural networks	✓		✓	✓	✓	
Adi et al. [42]	Deep neural networks		✓	✓			
Zhang et al. [43]	Deep neural networks		✓	✓			
Guo et al. [44]	Deep neural networks		✓		✓		
Fan et al. [39]	Deep neural networks	✓		✓			
Namba et al. [45]	Deep neural networks		✓	✓			
DeepSigns [46]	Deep neural networks	✓	✓		✓		
Sakazawa et al. [47]	Deep neural networks		✓	✓			
Li et al. [48]	Deep neural networks		✓	✓			
Le Merrer et al. [49]	Deep neural networks		✓	✓			
BlackMarks [50]	Deep neural networks		✓		✓		
Fingerprinting							
DeepMarks [51]	Machine learning	✓			✓		
IPGuard [52]	Deep neural networks	✓					✓

use them as watermark; include data samples not related to the current task during the training phase and use them as watermark; include noise during the training phase and use it as watermark. The key for the watermark is composed by the pairs patterns, predictions that are known by the legitimate owner of the deep neural network. The verification of the watermark is carried out by performing normal requests to the deep neural network using the watermarking patterns as inputs and matching the results with the expected predictions. The proposed approach is evaluated using benchmark data sets taken from image processing, and the watermarking technique is verified to be robust against classical watermark removal attacks including fine-tuning, parameter pruning and model inversion attacks.

A similar approach to that of Zhang et al. is proposed by Li et al. [48]. The proposed framework uses an encoder to embed a watermark in the dataset. The encoder combines the regular input dataset with a 'logo', and is targeted at making the final watermarked images barely distinguishable from the original images. The discriminator is trained to distinguish between training samples and 'trigger' samples embedded with the watermark. The model is trained so that a specific label is assigned to watermarked inputs – and this backdoor provides the basis for model ownership claims.

The work by Sakazawa et al. [47] introduces a novel idea for watermark detection: the cumulative decoding of the embedded watermarks. The embedded patterns are cumulatively (and visually) decoded using only a subset of the key watermark dataset, which enables third-party verification without requiring disclosure of the entire dataset. In their experiments, 20 images out of 60,000 are used to decode the watermark. This translates into the fact that means only a subset of watermark 'key' images are enough to decode, without significant performance degradation to the original image classification task.

The principle of known pairs is also used by Guo and Potkonjak [44] in their black-box approach for watermarking a deep neural network in embedded systems. The watermark is embedded during the fine-tuning phase. The deep neural network is trained using the original data set and a modified dataset in which data are altered according the watermark signature. During normal operation, the deep neural network behaves normally. When an input belonging to the signature dataset is encountered, the deep neural network behaves in a predefined way. Reported results obtained using datasets from image classification showed that the approach incurs a limited performance degradation, while confirming that the proposed watermark could be embedded into the considered deep neural network models.

Fan et al. [39] proposed a passport-based scheme for the verification of a deep neural network's ownership. The passport-based approach is imposed by the need to mitigating so-called ambiguity attacks, which aim at creating confusion during watermark verification (we will describe them in detail in Section 5.4). The main idea presented in the study consists of two steps. First, the passport is included during the design and the training of the deep neural network models, such as to make both of them dependent on the passport. Second the verification process is expanded to consider, as part of the signature verification step, both the behaviour of the deep neural network models when predefined inputs are applied and the performance of the models themselves. If a wrong

passport is used, then the performance of the model is different than the expected one. Practically, the passports are embedded by appending, after each convolution layer, a passport layer. The specific parameters of the passport, including bias and scale factors, are selected depending also on the weight of the kernel after which the passport is appended. The passport-based approach presented requires the access to internals of the model (thus belonging to white-box setting). However, the authors propose to use it in combination with other black-box approaches, to maintain the advantages of black-box approaches, while mitigating the effects of ambiguity attacks complementing the used black-box approach with the passport-based watermarking technique. The authors verify the proposed approach using classical benchmarks and release the code and the model used to carry out the experiments.

Namba and Sakuma [45] present a watermarking method called exponential weighting. The proposed method aims at defeating attacks that intercept the queries used to verify the authorship of the models and modify the images removing the verification tag (more details about query modification attacks are given in Subsection 5.3). This attack is successful because it is possible to identify images used for verification since they are often regular images with a certain superimposed signing feature (such as a specific word or a logo). To mitigate this attack and being resistant against model modifications, the authors propose a watermarking method that consists of two components. The first is to use verification samples that are not distinguishable from the regular training samples, and limit the change imposed by the watermark only to the labels. The second is to use a training algorithm in which only model parameters with large absolute values contribute to the predictions. The proposed approach is realized in the black-box settings, thus does not require access to the internal model to carry out the ownership verification step. The authors experimentally show the performance of the watermarking verification and its robustness against known attacks, including model modification and query verification. Reported results show that the performance of the model is marginally affected by the presence of the watermarks.

A formal framework for embedding watermarks in deep learning models has been proposed by Rouhani et al. [46]. Practically, DeepSigns, the name of the framework, is encapsulated as a high-level wrapper that can be used with state-of-the art deep learning frameworks. The framework includes both white-box and black-box settings. The watermarks are embedded in the probability density distribution of the activation sets corresponding to different layers of the neural network to be protected. Depending on the settings, the watermark can be embedded in the hidden layers of the model, or in the output layer. Reported results on several state-of-the-art datasets showed the practicability of the approach and the resistance against compression attacks, fine-tuning attacks and watermark overwriting attacks.

Merrer et al. [49], following the principle used during adversarial attacks, propose to add a small perturbation to a number of original samples so that they are classified incorrectly, and then to correct the labels so that they are labelled correctly. The model with corrected labels is then used as watermark. Since models without the watermark are likely to not correctly classify the adversarial samples, the ownership verification can be carried out by measuring the gap between the correctly classified adversarial examples and the incorrectly classified ones. The proposed approach is suitable for operation in the black-box setting, and the embedding is performed during the training phase. Reported experiments show good performance of the watermark, even if some false positive during the watermark extraction phase would require further analysis to be explained in depth.

Still in the black-box setting, Chen et al. [50] propose the framework BackMarks to watermark deep neural networks. The embedding of the proposed watermark requires few steps. The first step is the generation of a model-dependent encoding scheme that splits all the classes belonging to a specific task into two groups. The second step, that requires the binary string corresponding to the signature of the model owner, is the generation of the watermark key pairs using adversarial attacks. These watermarks are finally embedded in the models of the deep neural network in the fine-tuning phase. The signature is extracted using the key pairs that return the encoded signature of the model owner. The signature is finally decoded and verified. The proposed approach is applied to state-of-the-art datasets. Reported results show a runtime overhead of approximately 2% for embedding the watermark, while the functionality of the original deep neural network is maintained.

Improvements in the way in which the patterns to trigger the watermarks have been also studied. Guo and Potkonjak [54] proposed to use an evolutionary algorithm to generate trigger patterns that incur in a limited false-positive rate while maintaining the robustness of the embedded watermark against state-of-the-art attacks.

4.2 | Fingerprinting

Fingerprinting is less explored than watermarking. Chen et al. [51] proposed the framework DeepMarks to allow the owner of deep learning models to embed a unique fingerprint within the weights of the model itself. The fingerprinting is performed by assigning a unique binary code vector to each user of the mode. As in the case of DeepSign, the unique identifier is encoded and embedded in the probabilistic distribution on the weights. The proposed methodology is verified using state-of-the-art datasets. Reported results show accuracy comparable to the one of the baseline neural network, while achieving resistance against classical attacks such as fingerprint collusion, parameter pruning and model fine-tuning.

Fingerprinting can be combined with trusted execution environment to provide a complete attestation method for AI. In the approach proposed by Chen et al. [55], the fingerprint embedded during the fine-tune of the deep neural network is extracted with the support of the trusted execution environment and used as attestation criteria to ensure that only the programs that can correctly match the fingerprint are allowed to be used in the target hardware device. The attestation

support is deployed with the support of high-level APIs for integrating it into existing deep learning frameworks.

A recent work in progress (Cao et al. [52]) proposes to fingerprint a specific deep neural network model instead of fingerprinting each user, and to use this fingerprint instead of the watermark to detect IP violation. The main assumption behind this approach is that a deep neural network classifier can be uniquely represented by its classification boundaries. The fingerprint is then built by carefully selecting some points across this boundary. The verification is carried out by querying these points on a suspected model. If the labels predicted by the suspected model are the same as the ones predicted by the classifier of the model owner, the suspected model is considered to be the copied. An extensive evaluation of the security and suitability of this approach is however still under process.

5 | ATTACKS

In this section, we analyze the different attacks that have been proposed in the study against watermarking and fingerprinting of ML models. Interestingly, Quiring et al. note how the ML community and the watermarking community have independently developed similar attack and defense strategies [56]. The attacks on ML watermarking are therefore quite similar in concept and application to the larger class of attacks on ML. For the scope of this survey, however, we only consider attacks that have specifically been designed and evaluated against watermarking and fingerprinting of ML.

In Table 2, we summarize which watermarking and fingerprinting schemes are affected by the surveyed attacks.

5.1 | Model transformations

Given a trained model, transformation techniques enable deriving a new model with the same or slightly different prediction task. This is normally done to embed additional features such as memory and computational efficiency, however, transformation can be used to effectively eliminate the watermark. This strategy is also referred to as watermark suppression, as it aims at preventing the watermark detection. Yang et al. [57] provide list of model transformations that can be used for this purpose, and evaluate their effectiveness against a number of watermarking schemes. We list and briefly describe the transformations below, while their effectiveness against the surveyed schemes is summarized in Table 2. As this family of attacks can be applied against any trained model, they are independent of the watermarking technique used. In the table, we report a high level of effectiveness when the attack results in an accuracy of watermark extraction from the model below 0.25, where 1 is a perfectly preserved watermark and 0 is a watermark that has been fully removed from the model. When no reference is provided, the effectiveness results are from the proposer of the respective watermarking scheme.

In general, the most common type of model transformation is compression, where the objective is to optimize the memory needed to fit the (parameters of the) model, while preserving its accuracy. This technique, originally introduced to optimize the network, is abused by the adversaries to suppress an embedded watermark. Distillation [41] is a compression technique that aims to 'distill' the original model's knowledge into another model of smaller size, for example by reducing the number of neurons in each layer. By effectively retraining a new model, this technique is very effective in removing watermarks: this is due to the fact that distillation does not retain the watermark embedded in the model, as long as it is redundant and independent to the main learning task. To counter this, Yang et al. [57] propose a technique which forces the model to also represent the knowledge of the watermark.

Pruning [58] and rounding [58] are two simpler compression mechanisms, that instead act on the parameters: in pruning, the compression is achieved by removing insignificant parameters and pruning their links between neurons (often followed by fine-tuning the remaining parameters); in rounding, by reducing the precision of the parameters, limiting the number of required bits to represent them. They are partially effective in removing watermarking embedded into the statistical information of parameters (such as Uchida et al. [40]), but are ineffective when the embedding is achieved through capacity abuse (e.g. Adi et al. [42]).

A very common practice in ML is to refine and update a model using new data. This process of retraining a model using a refining set is normally referred to as fine-tuning [64]. Similarly to compression, fine-tuning can be targeted at removing watermarking [65]. In particular, Chen et al. [59] propose a carefully designed fine-tuning method which enables the adversaries to effectively remove watermarks embedded using pattern-based (e.g. [43]) and instance-based techniques (e.g. [42]), without compromising the model functionality.

5.2 | Watermark removal

As the name implies, watermark removal attacks aim at removing the watermark from an IP. Variations of watermark removal also exist: watermark overwriting [46], where the adversary attempts, at the same time, to remove the existing watermark and to insert a new one; and watermark forging, where the adversary tries to maliciously claim ownership of the watermark in the stolen model [43]. In the following we discuss current attack based on these strategies.

Wang and Kerschbaum [60] proposed a first attack against the original Uchida et al. watermark design [40]. The attack reliably detects and removes the digital watermarks by exploiting the model parameter distribution variations introduced by the Uchida et al. algorithm, notably the weights standard deviation.

Shafieinejad et al. [61] present a comprehensive suite of attacks aimed at backdoor-based schemes [42–44]. In particular, they propose a black-box attack, a white-box attack and a property inference attack. The first two attacks do not require access to the trigger set, the ground truth function of the

TABLE 2 The schemes targeted by attacks in the literature. For model transformation attacks, which can be applied to any trained model, we use the results of Yang et al. [57] in classifying the effectiveness: high, low, none ('-') or unknown ('?'). When no reference is provided, the results are from the proposer of the respective watermarking scheme. For targeted watermarking removal attacks, the table has been compiled using the results of the authors of the attacks. An 'x' indicates the attack is successful against the scheme. These schemes are not applicable to fingerprinting

	Targeted model transformations				**Watermarking removal**			
Schemes	**Distillation [41]**	**Pruning [58]**	**Rounding [58]**	**Fine-tuning [59]**	**Wang et al. [60]**	**Shafieinejad et al. [61]**	**REFIT [62]**	**Aiken et al. [63]**
Watermarking	Uchida et al. [40]	High [57]	Low [57]	Low [57]	Low [57]	x		
Adi et al. [42]	High [57]	- [57]	- [57]	High [59]		X	x	x
Zhang et al. [43]	?	High [45]	?	High [59]		X	x	x
Guo et al. [44]	?	?	?	?		X		
Fan et al. [39]	?	Low	?	-				
Namba et al. [45]	?	-	?	?			x	
DeepSigns [46]	?	High [45]	?	-				
Sakazawa et al. [47]	?	?	?	?				
Li et al. [48]	?	?	?	?				
Le Merrer et al. [49]	?	Low [45]	?	Low (IRNN)			x	
BlackMarks [50]	?	-	?	-				
Fingerprinting								
DeepMarks [51]	?	-	?	Low				
IPGuard [52]	?	?	?	?				

watermarked model or any labelled data. Instead, they use inputs from the publicly known domain of the watermarked model, and query the model for labels. Experimental results indicate a strong effectiveness of both strategies. The property inference attack, on the other hand, aims at distinguishing a watermarked model from an unmarked one. This can be used in two stages of the watermark removal process: as first step, to confirm the presence of the watermark, and as the last step, to confirm successful removal.

REFIT [62] is a framework for watermark removal that does not require knowledge of the watermarks or of the schema used to embed the watermark. The watermark removal is carried out during the fine-tune phase that is extended with two dedicated techniques: one derived from the elastic weight consolidation, and one exploiting the augmentation of unlabelled data. Reported preliminary results show the effectiveness of the approach in removing the embedded watermark also in scenarios were the adversary has limited training data.

Aiken et al. [63] propose a 'laundering' algorithm aiming to remove watermarks-based black-box methods ([42, 43]) using low-level manipulation of the neural network based on the relative activation of neurons. This effectively removes the watermark protection while retaining relatively high accuracies in the resulting models (between 80% and 97% in the study experiments). The authors conclude that current backdoor-based watermarks are overestimating the robustness of these watermarks while underestimating the ability of attackers to retain high test accuracy. These claims, supported by their experimental results, seem validated by the similar results by Shafieinejad et al. [61], and indicate that significant improvement is needed for the second generation of backdoor-based watermarks.

5.3 | Query modification

Query modification [45] is an attack that aims at altering the query used to verify the ownership of an AI model to render the whole verification process ineffective. Often, the data used to carry out the owner verification process are regular data with superimposed strings (such as a logo or a 'test' text) on top. Because of this, it is possible to determine whether a query belongs to the verification process or not. Once the query is identified by the attacker, the data used for the verification are modified, removing the superimposed strings and thus reverting back to the original format. The reverted data are then classified correctly (thus not returning the verification string), and the verification process fails.

5.4 | Ambiguity attacks

Ambiguity attacks are the final class of attack covered in this survey. Also these attacks have been previously analyzed in the context of digital image watermarking and then applied to deep neural networks [39]. At high level, the goal of an ambiguity attack is to create doubts about the ownership of an IP by embedding into the protected IP itself a number of forged watermarks. Ambiguity attacks are effective if the false-positive rate of the watermarking schema is sufficiently high. The exact definition of 'sufficiently high' is strongly dependent on the application. However, it was reported in study that a false-positive probability of less than 10^{-10} could be needed [66] to significantly limit ambiguity attacks.

Ambiguity attacks against watermarks of AI implementations are the least studied class of attacks. Robustness of existing watermarking schemes against ambiguity attacks has been addressed only recently [39], and it was shown that at least one representative of each class of watermarking techniques is susceptible against these attacks (with limited effort for the attacker). An approach to mitigate these attacks has been previously described in Section 4.

6 | DISCUSSION AND CONCLUSIONS

AI algorithms and models have achieved outstanding results in many fields, including speech recognition, health care and automotive. To develop and train an AI algorithm is a very costly and time consuming process. As a result, the owner of the model should be able to protect it, avoiding misuse and piracy. We reviewed IP protection techniques applied in the field of AI.

Concerning the protections, the majority of the works are focused on watermarking, while very few are considering other functionalities such as fingerprinting or even more advanced schemes such as attestation. We believe thus that there is a need of a more in depth analysis of possible fingerprinting techniques for AI models. Furthermore we believe that it is important to explore solutions, also at architectural level, to provide enhanced functionalities for IP right management, following the initial works on attestation of deep neural network. Still on protections, several works proposed so far are designed for the black-box settings, thus have the advantage of not requiring access to internal data of the model. However, there are functionalities that cannot be implemented in this settings yet (for instance, resistance against ambiguity attacks). We suggest to explore the possibility of providing robustness against this attack while still operating in the black-box setting or, alternatively, to explore ways for efficiently combining white- and black-box settings. This last approach was only recently proposed and appears to be promising.

With regard to attacks, most schemes being proposed are evaluated against the basic pruning attack, which however has limited significance as a benchmark against more advanced and powerful attacks. Similarly, the schemes evaluated against fine-tuning normally use a trivial fine-tuning attack, and as such results do not fully reflect the potential capability of this approach, as demonstrated by Chen et al. [59] with their carefully designed attack. Among the model transformation attacks, distillation is the most promising, as demonstrated by Yang et al. [57], although results exist only for a limited number of schemes. We believe research in distillation attacks, how they impact current and future schemes and potential countermeasures should be a focus of future investigation.

ORCID

Francesco Regazzoni https://orcid.org/0000-0001-6385-0780

Paolo Palmieri https://orcid.org/0000-0002-9819-4880

REFERENCES

1. Copeland, B.: Artificial intelligence. Britannica (2020). https://www.britannica.com/technology/artificial-intelligence. Accessed 8 Jan 2021
2. Morgan, S.: Official annual cybercrime report. Herjavec Group (2019). https://www.herjavecgroup.com/resources/2019-official-annual-cybercrime-report/. Accessed 8 Jan 2021
3. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, (ICLR), Conference Track Proceedings, San Diego (2015). http://arxiv.org/abs/1412.6572
4. Sharif, M., et al.: Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: Weippl, E.R. et al. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1528–1540. ACM, Vienna (2016). https://doi.org/10.1145/2976749.2978392
5. Brown, T., et al.: Adversarial patch (2017). http://arxiv.org/abs/1712.09665
6. Eykholt, K., et al.: Robust physical-world attacks on deep learning visual classification. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, pp. 1625–1634. IEEE Computer Society, Salt Lake City (2018). http://openaccess.thecvf.com/content_cvpr_2018/html/Eykholt_Robust_Physical-World_Attacks_CVPR_2018_paper.html
7. Juuti, M., et al.: PRADA: protecting against DNN model stealing attacks. In: IEEE European Symposium on Security and Privacy, EuroS&P, pp. 512–527. IEEE, Stockholm (2019). https://doi.org/10.1109/EuroSP.2019.00044
8. Chakraborty, A., Mondal, A., Srivastava, A.: Hardware-assisted intellectual property protection of deep learning models. In: 57th ACM/IEEE Design Automation Conference, DAC, pp. 1–6. IEEE, San Francisco (2020). https://doi.org/10.1109/DAC18072.2020.9218651
9. Poole, D., Mackworth, A.K., Goebel, R.: Computational intelligence - a logical approach. Oxford University Press (1998)
10. Dick, S.: Machines who write [think piece]. IEEE Ann. Hist. Comput. 35(2), 88 (2013). https://doi.org/10.1109/MAHC.2013.21
11. Tan, C.J.: Deep blue: Computer chess and massively parallel systems (extended abstract). In: Valero, M. (ed.) Proceedings of the 9th international conference on Supercomputing, ICS, pp. 237–239. ACM, Barcelona (1995). https://doi.org/10.1145/224538.224566
12. Chen, J.X.: The evolution of computing: Alphago. Comput. Sci. Eng. 18(4), 4–7 (2016). https://doi.org/10.1109/MCSE.2016.74
13. Walsh, V., Taylor, H.R.: Automatic speech recognition using artificial intelligence methods. In: European Conference on Speech Technology, ECST, pp. 1108–1111. ISCA, Edinburgh (1987). http://www.isca-speech.org/archive/ecst_1987/e87_1108.html
14. Skinner, D.R., Benke, K.K.: A machine vision system with learning capabilities. In: Barter, C.J., Brooks, M.J. (eds.) AI '88: 2nd Australian Joint Artificial Intelligence Conference, Proceedings.of Lecture Notes in Computer Science, vol. 406, pp. 328–346. Springer, Adelaide (1988). https://doi.org/10.1007/3-540-52062-7_88
15. Wang, T., et al.: Artificial intelligence for vehicleto-everything: a survey. IEEE Access. 7, 10823–10843 (2019). https://doi.org/10.1109/ACCESS.2019.2891073
16. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine learning, neural and statistical classification. Imprint of Simon and Schuster One Lake Street Upper Saddle River (1994). Accessed 23 March 2021
17. Alpaydin, E.: Introduction to machine learning, Adaptive computation and machine learning, 4th edn. MIT Press, Cambridge (2020)
18. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. 3, 95–99 (1988)
19. Xu, R., II, D.C.W.: Survey of clustering algorithms. IEEE Trans. Neural Networks. 16(3), 645–678 (2005). https://doi.org/10.1109/TNN.2005.845141
20. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. J. Artif. Intell. Res. 4, 237–285 (1996). https://doi.org/10.1613/jair.301
21. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature. 521(7553), 436–444 (2015)
22. Schulz, H., Behnke, S.: Deep learning - layer-wise learning of feature hierarchies. Künstliche Intelligenz. 26(4), 357–363 (2012). https://doi.org/10.1007/s13218-012-0198-z
23. Kennedy, J.: Swarm intelligence. In: Zomaya, A.Y. (ed.) Handbook of nature inspired and innovative computing - integrating classical models with emerging technologies, pp. 187–219. Springer (2006). https://doi.org/10.1007/0-387-27705-6_6
24. Deneubourg, J.L., et al.: The self-organizing exploratory pattern of the argentine ant. J. Insect. Behav.. 3(2), 159–168 (1990)
25. Beni, G., Wang, J.: Swarm intelligence in cellular robotic systems. In: Robots and biological systems: towards a new bionics?, pp. 703–712. Springer (1993)
26. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of International Conference on Neural Networks (ICNN'95), pp. 1942–1948. IEEE, Perth (1995). https://doi.org/10.1109/ICNN.1995.488968
27. Dorigo, M.: Optimization, learning and natural algorithms. PhD Thesis. Politecnico di Milano (1992)
28. Hoh, C., Hwang, R.: P2P file sharing system over MANET based on swarm intelligence: A cross-layer design. In: IEEE Wireless Communications and Networking Conference, WCNC, pp. 2674–2679. IEEE, Hong Kong (2007). https://doi.org/10.1109/WCNC.2007.497
29. Hayes-Roth, F., Waterman, D.A., Lenat, D.B.: Building expert systems of Advanced book program, vol. 1. Addison-Wesley Longman Publishing Co., Boston (1983)
30. Reid, S.: Knowledge-based systems concepts, techniques, examples. Can. High Technol. 3(22), 238–281 (1985)
31. Puppe, F.: Systematic introduction to expert systems. knowledge representations and problem-solving methods. Springer-Verlag, Berlin; Heidelberg (1993)
32. Spyropoulos, C.D.: AI planning and scheduling in the medical hospital environment. Artif. Intell. Medicine. 20(2), 101–111 (2000). https://doi.org/10.1016/S0933-3657(00)00059-2
33. McDermott, D.V., Hendler, J.A.: Planning: What it is, what it could be, an introduction to the special issue on planning and scheduling. Artif. Intell. 76(1–2), 1–16 (1995). https://doi.org/10.1016/0004-3702(95)00034-C
34. desJardins, M., et al.: A survey of research in distributed, continual planning. AI Mag. 20(4), 13–22 (1999). http://www.aaai.org/ojs/index.php/aimagazine/article/view/1475
35. Paszke, A., et al.: Automatic differentiation in pytorch. In: 31st Conference on Neural Information Processing Systems (NIPS), Long Beach (2017). https://openreview.net/forum?id=BJJsrmfCZ
36. Jia, Y., et al.: Convolutional architecture for fast feature embedding. In: Hua, K.A. (ed.), et al. (eds.) Proceedings of the ACM International Conference on Multimedia, MM '14, pp. 675–678. ACM, Orlando (2014). https://doi.org/10.1145/2647868.2654889
37. Kahng, A. B., et al.: Watermarking techniques for intellectual property protection. In: Chawla, B.R., Bryant, R.E., Rabaey, J.M. (eds.) Proceedings of the 35th Conference on Design Automation, pp. 776–781. Moscone center, ACM Press, San Francico (1998). https://doi.org/10.1145/277044.277240
38. Dunbar, C., Qu, G.: A practical circuit fingerprinting method utilizing observability don't care conditions. In: Proceedings of the 52nd Annual Design Automation Conference, pp. 113:1–113:6. ACM, San Francisco (2015). https://doi.org/10.1145/2744769.2744780
39. Fan, L., Ng, K., Chan, C.S.: Rethinking deep neural network ownership verification: embedding passports to defeat ambiguity attacks. In: Annual Conference on Neural Information Processing Systems 2019, NeurIPS,

pp. 4716–4725. Vancouver, Canada (2019). https://proceedings.neurips.cc/paper/2019/hash/75455e062929d32a333868084286bb68-Abstract.html
40. Uchida, Y., et al.: Embedding watermarks into deep neural networks. In: Ionescu, B. (ed.), et al. (eds.) Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR, pp. 269–277. ACM, Bucharest (2017). https://doi.org/10.1145/3078971.3078974
41. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR (2015). http://arxiv.org/abs/1503.02531
42. Adi, Y., et al.: Turning your weakness into a strength: watermarking deep neural networks by backdooring. In: Enck, W., Felt, A.P. (eds.) 27th USENIX Security Symposium, USENIX Security, pp. 1615–1631. USENIX Association, Baltimore (2018). https://www.usenix.org/conference/usenixsecurity18/presentation/adi
43. Zhang, J., et al.: Protecting intellectual property of deep neural networks with watermarking. In: Kim, J. (ed.), et al. (eds.) Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS, pp. 159–172. ACM, Incheon (2018). https://doi.org/10.1145/3196494.3196550
44. Guo, J., Potkonjak, M.: Watermarking deep neural networks for embedded systems. In: Bahar, I. (ed.) Proceedings of the international conference on computer-aided design. ICCAD, pp. 133. ACM, San Diego (2018). https://doi.org/10.1145/3240765.3240862
45. Namba, R., Sakuma, J.: Robust watermarking of neural network with exponential weighting. In: Galbraith, S. D. (ed.), et al. (eds.) Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS, pp. 228–240. ACM, Auckland (2019). https://doi.org/10.1145/3321705.3329808
46. Rouhani, B.D., et al.: An end-to-end watermarking framework for ownership protection of deep neural networks. In: Bahar, I., et al. (eds.) Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS, pp. 485–497. ACM, Providence (2019). https://doi.org/10.1145/3297858.3304051
47. Sakazawa, S., et al.: Visual decoding of hidden watermark in trained deep neural network. In: 2nd IEEE Conference on Multimedia Information Processing and Retrieval, MIPR, pp. 371–374. IEEE, San Jose (2019). https://doi.org/10.1109/MIPR.2019.00073
48. Li, Z., et al.: How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN. In: Balenson, D. (ed.) Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC, pp. 126–137. ACM, San Juan (2019). https://doi.org/10.1145/3359789.3359801
49. Merrer, E.L., Pérez, P., Trédan, G.: Adversarial frontier stitching for remote neural network watermarking. Neural Comput. Appl. 32(13), 9233–9244 (2020). https://doi.org/10.1007/s00521-019-04434-z
50. Chen, H., Rouhani, B.D., Blackmarks, F.K.: Blackbox multibit watermarking for deep neural networks. CoRR (2019). http://arxiv.org/abs/1904.00344. Accessed 23 March 2021
51. Chen, H., et al.: A secure fingerprinting framework for digital rights management of deep learning models. In: El-Saddik, A. et al. (eds.) Proceedings of the 2019 on International Conference on Multimedia Retrieval, ICMR, pp. 105–113. ACM, Ottawa (2019). https://doi.org/10.1145/3323873.3325042
52. Cao, X., Jia, J., Gong, N.Z.: IPGuard: Protecting the intellectual property of deep neural networks via fingerprinting the classification boundary. CoRR (2019). http://arxiv.org/abs/1910.12903
53. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) In: 3rd International conference on learning representations, ICLR 2015. Conference Track Proceedings, San Diego (2015). http://arxiv.org/abs/1412.6572.May 7-9
54. Guo, J., Potkonjak, M.: Evolutionary trigger set generation for DNN black-box watermarking. CoRR (2019). http://arxiv.org/abs/1906.04411
55. Chen, H., et al.: Deepattest: an end-toend attestation framework for deep neural networks. In: Manne, S.B. (ed.) Proceedings of the 46th International Symposium on Computer Architecture, ISCA, pp. 487–498. ACM, Phoenix (2019). https://doi.org/10.1145/3307650.3322251
56. Quiring, E., Arp, D., Rieck, K.: Forgotten siblings: Unifying attacks on machine learning and digital watermarking. In: 2018 IEEE European Symposium on Security and Privacy, EuroS&P, pp. 488–502. IEEE, London (2018). https://doi.org/10.1109/EuroSP.2018.00041
57. Yang, Z., Dang, H., Chang, E.: Effectiveness of distillation attack and countermeasure on neural network watermarking. CoRR (2019). http://arxiv.org/abs/1906.06046
58. Han, S., Mao, H., Dally, W.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR, Conference Track Proceedings. San Juan (2016). http://arxiv.org/abs/1510.00149
59. Chen, X., et al.: Leveraging unlabeled data for watermark removal of deep neural networks. In: ICML Workshop on Security and Privacy of Machine Learning, Long Beach (2019)
60. Wang, T., Kerschbaum, F.: Attacks on digital watermarks for deep neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, pp. 2622–2626. IEEE, Brighton (2019). https://doi.org/10.1109/ICASSP.2019.8682202
61. Shafieinejad, M., et al.: On the robustness of the backdoor-based watermarking in deep neural networks. CoRR (2019). http://arxiv.org/abs/1906.07745
62. Chen, X., et al.: REFIT: a unified watermark removal framework for deep learning systems with limited data. CoRR (2019). http://arxiv.org/abs/1911.07205
63. Aiken, W., Kim, H., Woo, S.S.: Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. CoRR (2020). https://arxiv.org/abs/2004.11368
64. Girshick, R., et al.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 580–587. IEEE, Columbus Computer Society (2014). https://doi.org/10.1109/CVPR.2014.81
65. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: Bailey, M., et al. (eds.) Research in Attacks, Intrusions, and Defenses- 21st International Symposium Proceedings of Lecture Notes in Computer Science, vol. 11050, pp. 273–294. RAIDSpringer, Heraklion (2018). https://doi.org/10.1007/978-3-030-00470-5_13
66. Sencar, H.T., Memon, N.: Combatting ambiguity attacks via selective detection of embedded watermarks. IEEE Trans. Inf. Forensics Secur. 2(4), 664–682 (2007)

How to cite this article: Regazzoni F, Palmieri P, Smailbegovic F, Cammarota R, Polian I. Protecting artificial intelligence IPs: A survey of watermarking and fingerprinting for machine learning. *CAAI Trans. Intell. Technol.* 2021;6:180–191. https://doi.org/10.1049/cit2.12029