Identifying Appropriate Intellectual Property Protection Mechanisms for Machine Learning Models: A Systematization of Watermarking, Fingerprinting, Model Access, and Attacks

Isabell Lederer, Rudolf Mayer, and Andreas Rauber

***Abstract*— The commercial use of machine learning (ML) is spreading; at the same time, ML models are becoming more complex and more expensive to train, which makes intellectual property protection (IPP) of trained models a pressing issue. Unlike other domains that can build on a solid understanding of the threats, attacks, and defenses available to protect their IP, ML-related research in this regard is still very fragmented. This is also due to a missing unified view as well as a common taxonomy of these aspects. In this article, we systematize our findings on IPP in ML while focusing on threats and attacks identified and defenses proposed at the time of writing. We develop a comprehensive threat model for IP in ML, categorizing attacks and defenses within a unified and consolidated taxonomy, thus bridging research from both the ML and security communities.**

***Index Terms*— Attacks on intellectual property protection (IPP), fingerprinting, IPP, machine learning (ML), model access control, watermarking.**

I. Introduction

IN MANY machine learning (ML) settings, training an effective model from scratch—especially complex and powerful models such as a deep neural network (DNN)—is computationally very expensive and requires expertise for setting parameters, and the amount of data needed is not commonly accessible or expensive to obtain. Security concerns become more prominent when these models are made available to other parties or customers, e.g., in ML as a Service (MLaaS) or under a license. This is when model owners—who have invested significant resources to train a model and now want to offer it to customers—start to consider intellectual property protection (IPP) methods such as watermarking (to verify ownership) and access control (to prevent unauthorized usage of a model). IP litigation cases over ML models do occur but have so far not seen widespread media attention; however, protection mechanisms are therefore investigated from a legal point of view, e.g., [1], showing that the burden of proof generally lies with the IP rights holder. Thus, it is important to anticipate the need for such proofs and protect ML models with IPP techniques.

In the last few years, we have consequently seen an increase in research on IPP techniques for ML models. Many black- and white-box watermarking methods have been proposed based on techniques such as backdoor embedding via data poisoning or regularization. At the same time, several studies have shown the vulnerability of some of these schemes against novel attacks. Similar observations hold true for model access control techniques. However, a comprehensive overview on the field, including a unified nomenclature and taxonomy, is still missing. Based on a systematic review, this article provides a survey and systematization of knowledge.

Our contributions are given as follows:

1) a systematic overview on research related to IPP of ML, focusing on reactive (watermarking and fingerprinting) and proactive (e.g., model access) techniques;
2) a taxonomy to categorize ML IPP schemes;
3) a categorization of 36 approaches by a set of characteristics identified through methodological comparison;
4) an analysis of vulnerability to attacks designed to break the IPP schemes;
5) guidelines on how to choose a fitting watermarking/IPP scheme for a given setting;
6) a framework for implementations of watermarking methods and available trained and watermarked models, allowing to compare other methods to previous research.[1]

The remainder of this article is structured as follows. Section II provides an overview of related surveys. Our research methodology is described in Section III. Section IV provides definitions and background to ML, DNNs, watermarking, and fingerprinting. Section V introduces our

Manuscript received 3 May 2021; revised 31 October 2021, 5 April 2022, and 3 August 2022; accepted 4 October 2022. This work was partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 826078 (project 'FeatureCloud'). This publication reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains. SBA Research (SBA-K1) is a COMET Center within the COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMAW, and the federal state of Vienna. The COMET Programme is managed by FFG. *(Corresponding author: Rudolf Mayer.)*

Isabell Lederer was with SBA Research, 1040 Vienna, Austria.

Rudolf Mayer and Andreas Rauber are with SBA Research, 1040 Vienna, Austria, and also with the Institute of Information Systems Engineering, Faculty of Informatics, Vienna University of Technology, 1040 Vienna, Austria (e-mail: rmayer@sba-research.org; andreas.rauber@tuwien.ac.at).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3270135.

Digital Object Identifier 10.1109/TNNLS.2023.3270135

[1] Available at https://sbaresearch.github.io/model-watermarking/

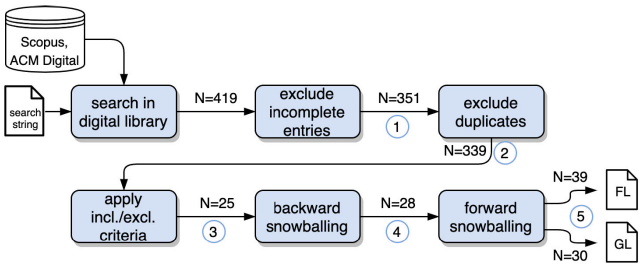

Fig. 1. Literature search process workflow. In every step, we denote the number of publications by $N = x$. Numbers 1–6 correspond to CSV-files, which contain all the retrieved literature in the particular step.

taxonomy of IPP methods, the threat model, and attacks. Sections VI and VII then discuss current approaches for watermarking and fingerprinting schemes, while Section VIII discusses proactive IPP methods such as access control. Section IX provides a taxonomy of currently known attacks and which IPP methods are vulnerable to them. In Section X, we provide guidelines for choosing fitting IPP methods in various scenarios before we provide our conclusions in Section XI.

II. Related Work

As the first work in this field, Chen et al. [2] empirically investigated five watermarking schemes for ML models (two white and three black boxes), evaluated their fidelity, and estimated the robustness against three attacks (model fine-tuning, parameter pruning, and watermark overwriting), thus providing an important early comparison of these techniques' effectiveness. We expand on this and provide a survey as well as systematization of the overall IP protection field for ML models. Concurrently to our work, a survey specifically covering the watermarking of ML models was published as a preprint by Boenisch [3]. Watermarking is an important aspect, which our work complements with fingerprinting and proactive methods such as model access, thus providing a holistic view of the entire IPP field.

III. Methodology

A. Literature Search

In preparation for this systematization, we performed an extensive literature search following the guidelines by Kitchenham and Charters [4]. Fig. 1 shows our literature search process. We distinguish between the following types of publications: formal literature (FL), i.e., peer-reviewed literature such as book sections, conference papers, and journal articles; and gray literature (GL), i.e., literature that did not undergo a peer-reviewed process, for example, preprints (published, e.g., on arXiv, university repositories, personal websites, and so on).

Fig. 2 shows the distribution of publications regarding the different topics and literature types.[2] We can clearly see that most papers address watermarking; however, there are also a significant number of papers on attacks. Note that some publications include both a novel attack on a scheme and a novel watermarking scheme, which is immune to this attack.

[2]The topics will be explained in more detail in Sections VI–IX.

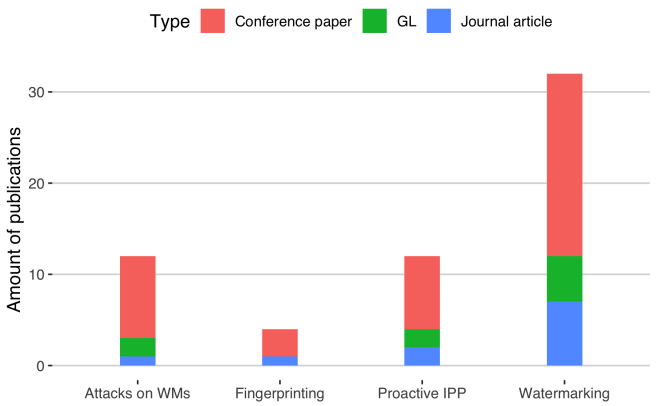

Fig. 2. Literature distribution across different topics regarding IPP of ML models. Most of the papers address watermarking.

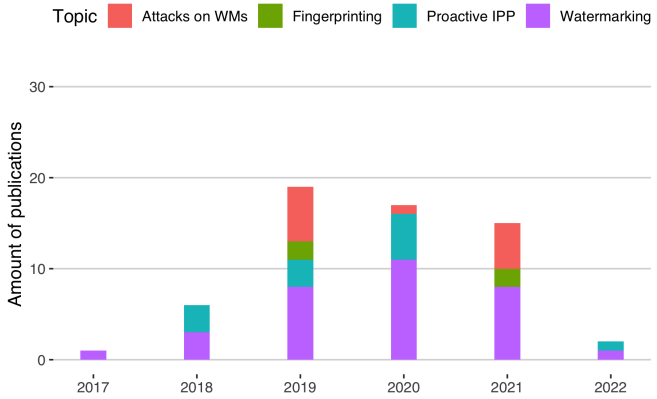

Fig. 3. Literature distribution over the years for different topics.

Fig. 3 shows the distribution of publications across the publishing years. We see a rising research interest in all topics, with papers on attacks being published only recently.

B. Inclusion and Exclusion Criteria

In order to facilitate the reproducibility of the literature search, we defined and documented the following criteria to find the most relevant literature covering IPP of ML models. Our inclusion criteria are: 1) literature that proposes an IPP scheme for ML models; 2) literature that proposes an attack on an IPP scheme for ML models; and 3) literature that evaluates or compares earlier schemes.

Our exclusion criteria are: 1) (near) duplicates[3]; (ii) literature that only uses ML for multimedia watermarking, such as image watermarking; and (iii) literature that only applies previously published IP protection schemes, without a novel or large-scale evaluation.

IV. Preliminaries

This section provides the necessary background for the remainder of this article. In this work, we focus on supervised learning, an area of ML including classification and regression.

[3]If the titles are different, but the content is very similar, we include all versions of this item and indicate that fact. However, we subsequently cite only the most complete version, as suggested by Kitchenham and Charters [4].

Some learning algorithms—such as the (stochastic) gradient descent commonly employed in DNNs or convolutional neural networks (CNNs)—iteratively adapt their learnable parameters by minimizing some form of loss function. To prevent overfitting to the training data, a parameter regularizer is oftentimes used. This is an additional term in the loss function, often in the form of a penalty that controls the magnitude of the parameter values. In the field of model IPP, regularizers are frequently used to embed a watermark into a model.

The process of fine-tuning, i.e., further training a model on different training data, usually with a smaller learning rate, can be used for either improving the model or, when using it for a slightly different purpose, in transfer learning. In the context of model IPP, we use it either to embed a watermark or as a malicious modification to a well-trained model to remove unwanted information (e.g., a watermark).

Knowledge distillation [5] is a compression technique that uses knowledge of a model (teacher network) to train a new, smaller, computationally cheaper model (student network).

Generative adversarial networks (GANs) use two models: one (the generator) learns the actual data generation task and the other (the discriminator) evaluates it [6].

Federated learning (FL) [7] is an ML technique in which multiple parties are involved to train the model on their data (without sharing the data, mostly to preserve privacy).

An autoencoder (AE) is a special artificial neural network that is commonly used for dimensionality reduction [8]. This is achieved by learning to replicate its input to its output via a smaller hidden layer that learns to represent the input.

Adversarial examples [9] are inputs created to fool a model. An original input is perturbed by some specially crafted noise such that the model is unable to classify the generated input instance correctly. The perturbation is kept minimal in order to be less noticeable by humans or technical detection methods.

We understand ML-based image processing as applying a model to an input image, with the output being an image as well. The model is trained to perform image enhancing and embed unrecognizable data or other transformations such as neural style transfer [10]. It is important to differentiate image processing from image preprocessing, which is usually performed on an image dataset before training a model and includes techniques such as resizing, cropping, or normalization.

A. Watermarking

Digital watermarking is a well-studied method, e.g., in multimedia [11] and relational databases [12] IPP. The main idea is to embed a piece of imperceptible[4] signature in the data (e.g., image or audio) to deter malicious usage. Digital watermarking is thus a form of steganography or information hiding, i.e., the practice of concealing a message within another message. The hidden information must be embedded in such a way that no algorithm can remove or overwrite the watermark. More recent digital watermarking techniques (e.g., for images) make use of deep learning techniques in the embedding process [13]; similarly, also attacks targeted to remove such watermarks are increasingly using deep learning [14]. Quiring and Rieck [15], for instance, combined methods from model stealing to generate a substitute model of a watermark detector and then generated adversarial examples against this model in order to obtain images with minimal perturbations, thus evading detection.

The watermarking methods we consider in this article are ML model watermarking, i.e., the IP that has to be protected is the ML model itself. Model watermarking is related to multimedia watermarking, but the techniques differ since the asset to be protected differs. Research on watermarking ML models predominantly addresses image classification (cf. Section VI) and, thus, CNNs. The introduced terminology is thus strongly influenced by this application of ML, but the concepts are transferable to other input types.

B. Fingerprinting

We consider fingerprinting as an extension of watermarking. While watermarking has the purpose to verify the owner of a digital asset, fingerprinting wants to trace its (potentially malicious) recipient. Therefore, fingerprinting techniques should be capable of embedding multiple, but unique marks to identify the recipient. Similar to watermarking, fingerprinting is already widely used in multimedia areas such as images, audio, video [16], or digital data stored in relational databases [17].

V. Taxonomy of IPP for ML Models

In this section, we define our threat model and provide a comprehensive taxonomy of IPP methods for ML models to mitigate the risks posed by those threats. Subsequently, we give an overview of attacks against those IPP mechanisms.

A. Threat Model

We first need to understand the motives of an attacker (also called adversary or malicious user). The entity that invested resources to obtain an ML model for a specific task ("model owner") wants to offer this model to a certain target audience/customer. The most prominent reasons for an attacker to (illegally) redistribute a model are: 1) no/not enough training data, expertise, time, or computational power to train such a model themselves and/or 2) the unwillingness to agree with the license terms of the obtained model or the fees for using MLaaS. We call the model to be protected the target model and the attacker's model—which stems from the target model—the adversary model. In our threat model, we assume one of the following scenarios.

1) *Legal Copy:* The model owner distributes the model publicly, either for free, e.g., via a platform such as Model Zoo [56], but with a restrictive license, or for a fee. The attacker redistributes it, e.g., via a lucrative API service.
2) *Illegal Copy:* The model owner distributes the model as a pay-per-query API service. The attacker performs

[4]Perceptible watermarks are also commonly used in the multimedia domain, e.g., logos or copyright notices superimposed on images or videos. Imperceptible watermarks, however, aim to avoid changing the perceptible impression of a work. This is the type of watermark we consider for IPP of ML models.

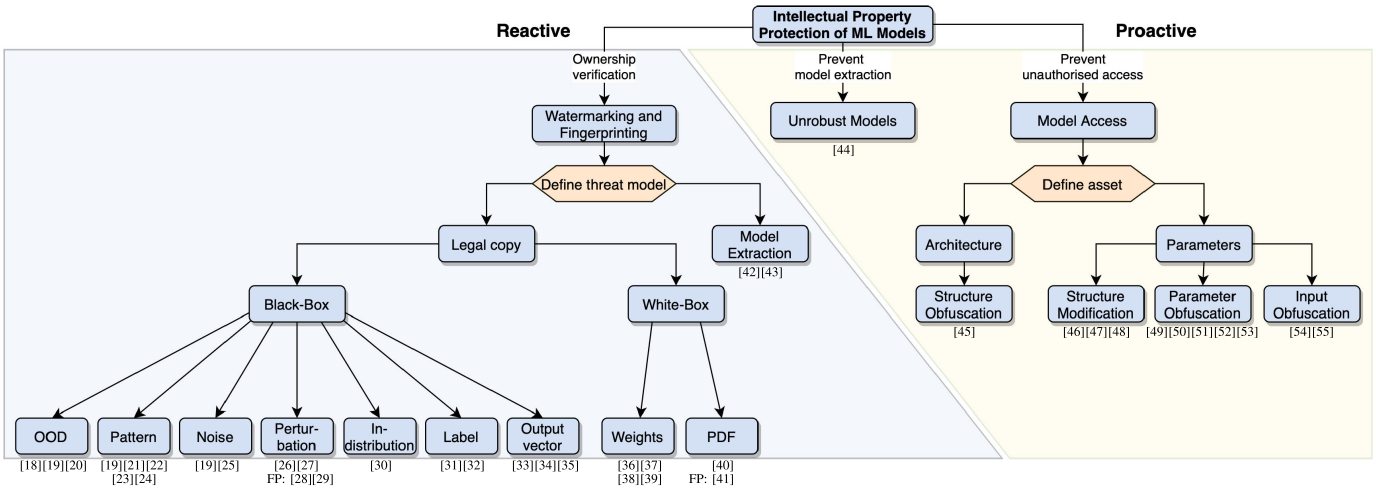

Fig. 4. Taxonomy of IPP mechanisms for ML models.

a model stealing (or extraction) attack [57], [58] and redistributes it as above, e.g., via their own API service.

Regardless of how the attacker obtained the model, in both cases, the IP of the model owner is illegally utilized. However, it is important to differentiate between those cases, as this has a large impact on the selection of potential defense mechanisms.

B. IPP Methods

We developed a comprehensive taxonomy of IPP methods for ML models, as shown in Fig. 4. We distinguish between reactive methods that respond to a threat event and proactive methods, meaning that the defender takes the initiative to prevent a threat event. Methods that enable to verify the ownership of a model through model watermarking and model fingerprinting are thus reactive; methods that, e.g., seek to prevent unauthorized model access are proactive. Ownership verification is a weak form of protection, as it requires the unauthorized usage of the model to be known (or at least suspected) and needs some form of access to the model. Model access control, on the other hand, may prevent such illegal use by rendering the model useless for unauthorized users. This is comparable to preventing unauthorized use of, e.g., software.

Some of the methods we introduce in this section can be distinguished by whether they are white-box or black-box. White box means that the model owner needs access to the parameters or other characteristics of the adversary model during the IPP method process, e.g., watermark extraction. As this scenario is often unrealistic, black-box mechanisms tend to be more popular. These generally only need access to the model's prediction—e.g., via an API service—to observe matching input and output from the ML model and use it in a similar fashion to an oracle.

Watermarking as defense against model stealing attacks (scenario 1) in our threat model) is mostly achieved through specific black-box watermarking techniques which survive such an attack, i.e., the hidden information is "stolen" with the model. In the case that a user legally obtained a copy of the ML model [scenario 2)], but then is using it not according to the licensing terms, more techniques are available. White-box approaches for this case embed the ownership information directly into the model parameters or their probability density function (pdf). Black-box approaches mostly rely on specific input samples, so-called "trigger sets," that will cause the model to behave in a way that is unexpected for the task and unpredictable to the attacker. These techniques mainly differ in how the respective triggers are constructed.

Model access control methods can be distinguished via the asset they want to protect. Most work focuses on the protection of the model parameters, either through encryption, other obfuscation techniques, or by requiring a specific method to transform the inputs. If the model structure (or architecture) is to be protected, usually obfuscation techniques are employed.

Watermarking and fingerprinting of ML models are forms of steganography (information hiding); however, we want to point out that there are several other connotations for watermarking, and information hiding in general, along the ML process (as shown in Fig. 5). For example, Sablayrolles et al. [59] proposed a technique that traces data usage; it marks (training) data so that an ML model trained on that data will bear a watermark that can be identified (cf. ① in Fig. 5). However, the main body of work regarding watermarking—and also the respective focus in this article—considers ML models as the asset to be protected through embedded watermarks (cf. ② in Fig. 5). Abdelnabi and Fritz [60] are not watermarking a model, but the output of a (text-)generating model (cf. ③ in Fig. 5). They assume that an attacker could use the model to generate entire articles; subsequently, the watermark can be extracted from the generated text and prove an illegitimate use of the model. For some settings, it is further considered that a marked output (prediction or data) is generated with the explicit goal to trace the usage of these data, e.g., by an attacker (cf. ④ in Fig. 5). This is a special form of ①, given that the data origin is different, and of ②, as the adversary model is implicitly marked (cf. Section VI-D).

Other forms of steganography may be employed in an attack against ML processes. For instance, Song et al. [63] proposed a technique to exfiltrate data from a private training dataset by hiding them within the parameters of a model that was

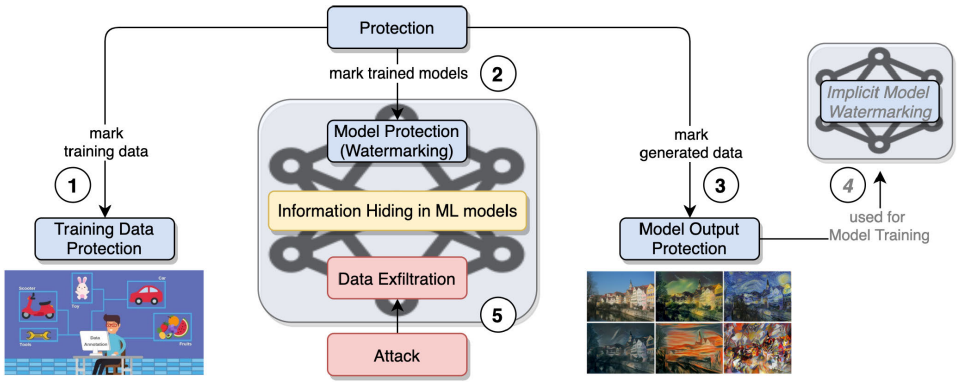

Fig. 5. Different notions of information hiding along an ML process.

trained on this dataset. In this way, adversaries who do not have direct access to the training data, but are allowed by the data owner to run an ML training algorithm on the data, can exfiltrate these data via the derived ML model, i.e., perform a data exfiltration attack (cf. (5) in Fig. 5).

In this work, however, we focus on techniques that hide and rightfully ingrain information about a legal owner or—in case of fingerprinting—a recipient of the model.

C. Attack Model

This section introduces the attack models, which we will further detail in Section IX. We assume that the attacker obtains a legal copy of the target model and either knows or suspects that the model has an IPP in place. We begin with attack models against watermarks, as these are transferable to other IPP types; we consider the following cases.

1) *Watermark Detection:* The attacker seeks to detect the watermark, potentially to perform a targeted watermark removal or overwriting. If the watermark is not secured with additional mechanism (e.g., a private key for extraction), the attacker could also claim ownership.
2) *Watermark Overwriting:* The attacker tries to replace the existing watermark with their watermark, thus rendering the model owner's watermark useless.
3) *Watermark Invalidation:* The attacker aims to disable the watermark function so that it cannot be verified, without actually removing it from the model.
4) *Watermark Removal:* The attacker wants to modify the model in such a way that the model owner's watermark extraction algorithm will no longer result in proving correct ownership.

Most of these attacks also work against fingerprinting.

With regard to model access control mechanisms, detection is often trivial, as an active mechanism will result in low fidelity. An attacker would mostly want to remove, invalidate, or potentially overwrite the mechanism to: 1) gain unauthorized access to use the model (as black box) or 2) to reveal either the model architecture or parameters for other purposes.

VI. Watermarking of ML Models

The vast majority of watermarking methods for ML models are designed specifically for DNNs. The main reason for this is not only the high value of DNNs, as they require large datasets and long training periods, but also the number of "degrees of freedom" in a DNN. Large DNNs thus have, compared to other ML models, more "space" for hiding watermarks. Most authors evaluate the schemes for image classification tasks. However, some extend to other tasks, such as audio classification [42], image processing [61], [62], [64] (the output being an image/data rather than a prediction), or specific settings, such as GANs [35], FL [24], graph neural networks [65], and deep reinforcement learning [66].

The following terminology is common in model watermarking and used throughout this article: watermark embedding is the process of placing the watermark into the model, e.g., via fine-tuning. Watermark extraction is the process of extracting the embedded watermark from the model, but neither in a malicious nor permanent way (which are called watermark detection and watermark removal). Extraction means to identify if and which watermark has been placed. Subsequently, during watermark verification, the extracted watermark is compared to the model owner's secret to prove ownership. Following certain rules, e.g., thresholding the bit error rate, it is determined if the watermarks are the same.

Typical workflows for white- and black-box watermarking are shown in Fig. 6(a) and (b), respectively. For white-box watermarking, the model owner creates a T-bit signature vector $\mathbf{b} \in \{0,1\}^T$, which is a set of arbitrary binary strings that should be independently and identically distributed (i.i.d.) [40]. This binary vector serves as a watermark and is usually embedded into the model through fine-tuning with regularization. We call this type of embedding scheme regularizer-based (cf. Section VI-B). Note that other ways of embedding are proposed by Uchida et al. [36], e.g., during the training or via knowledge distillation.

For black-box watermarking, the model owner creates specially crafted trigger inputs that receive wrong labels on purpose. When the model is "triggered" by these inputs, it behaves unexpectedly to a normal user (cf. Section VI-C).

A. Requirements

Watermarking (and fingerprinting) schemes should fulfill several requirements. Literature is not coherent in terminology; we therefore provide a common nomenclature in this article.

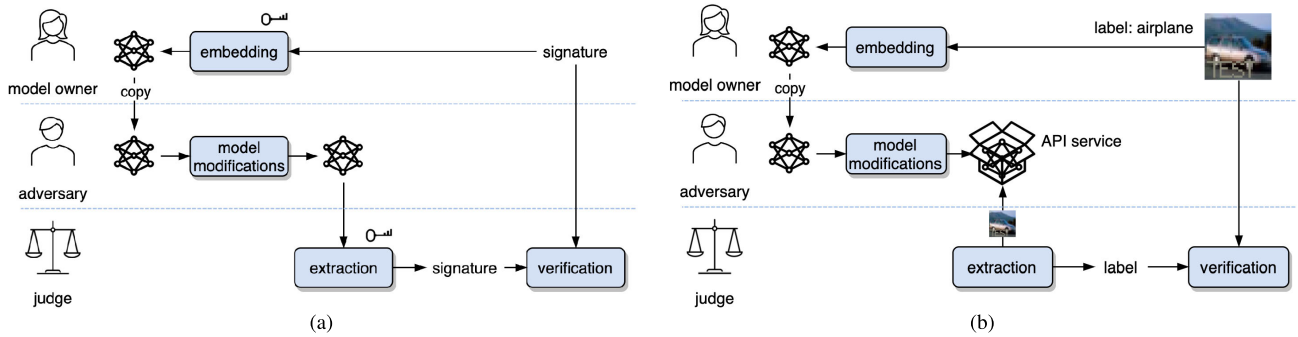

Fig. 6. Typical workflows for (a) white-box watermarking and (b) black-box watermarking.

TABLE I

REQUIREMENTS FOR WATERMARKING TECHNIQUES. THE NOTATION IS NOT CONSISTENT THROUGHOUT THE PAPERS, BUT THE TERMS IN THE LEFT COLUMN ARE THE MOST PROMINENT ONES. THESE REQUIREMENTS MOSTLY APPLY ALSO TO FINGERPRINTING METHODS

Property	Description	Other terms used in papers
Effectiveness	The model owner should be able to prove ownership anytime and multiple times if needed	Authentication [21], Functionality [27]
Fidelity	The accuracy of the model should not be degraded after embedding the watermark	Funcionality-preserving [18], [21], [37], Loyalty [26], Utility [43] (Image WM: Transparency [67]; relational data: Usability [12])
Robustness	The embedded watermark should resist a designated class of transformations	Unremovability [18], [43]
Security	The watermark should be secure against brute-force or specifically crafted evasion attacks	Secrecy [35], Unforgeability [18], [37]
Legality	An adversary cannot produce a watermark for a model that was already watermarked by the model owner	Ownership piracy resilient [18], [37], Non-ownership piracy [43]
Integrity	The watermark verification process should have a negligible false positive rate	Low false positive rate [22], [23], Non-trivial ownership [18], [21], [37], Uniqueness [64]
Reliability	The watermark verification process should have a negligible false negative rate	Credibility [33]
Efficiency	The watermarking embedding and verification process should be fast	
Capacity	The watermarking scheme should be capable of embedding a large amount of information	Payload [22]

To this end, we collected all requirements that are proposed in the literature and list them in Table I, identifying terms that are used synonymously and referencing respective publications.[5]

The most important requirements are effectiveness: the watermark should be embedded in a way that the model owner can prove ownership anytime; fidelity: the model's accuracy should not be degraded because of the watermark embedding; and robustness: the watermark embedding should be robust against several kinds of attacks, including fine-tuning, model compression, and other attacks specific to certain methods.

Fingerprinting should fulfill two more requirements, namely, uniqueness: the fingerprint can be uniquely attributed to a certain user and scalability: the fingerprinting scheme should be able to embed multiple fingerprints.

We provide an overview of all the watermarking and fingerprinting schemes considered in this article, and whether they are meeting the abovementioned requirements, in Table II. We observe that all schemes fulfill the above-identified most important requirements of fidelity, effectiveness, and robustness, except for Guan et al. [70] who purposefully give up robustness in favor of reversibility. This is inspired by traditional image integrity: the authors point out that the application of their scheme is not IPP, but integrity authentication, and that all existing watermarking methods are irreversible—once the watermark is embedded, it cannot be removed to restore the original model without degrading the model's performance. They argue that irreversible watermarking schemes alter the signature of a model, which could have severe consequences, especially in applications for, e.g., the medical or defense domain. The fidelity requirement does not apply for Zhang et al.'s method [61] since fidelity is not well-defined for generative models. As these output an image (or other complex data), whether a watermarked version of such a model is comparable to the original one requires defining an appropriate similarity measure to determine whether two outputs are equivalent.

B. White-Box Watermarking

White-box watermarking requires full access to the model during watermark extraction and verification.

The first framework for embedding a watermark into a DNN was proposed by Uchida et al. [36].[6] They follow the idea of embedding a signature into the model, particularly in the DNN's weights. Although it would be possible to directly alter

[5]Note that there are a few more terms used in the literature that cannot be easily mapped. Feasibility is a combination of robustness and effectiveness [27], and correctness of effectiveness, reliability, and integrity [29]. Nontrivial ownership is used in multiple ways—sometimes as a synonym for integrity, meaning that innocent models are not being accused of ownership piracy, but also as a requirement that an attacker cannot easily claim ownership without knowing the watermarking scheme and embedded watermark. Authentication is rather a subset of effectiveness than a real synonym since it only requires a provable association between an owner and their watermark.

[6]A slightly extended version can be found in [71].

TABLE II

REQUIREMENTS MET BY WATERMARKING AND FINGERPRINTING SCHEMES. WE DISTINGUISH TWO DEGREES. ∼: RESPECTIVE AUTHORS CLAIM THAT THE SCHEME FULFILLS THIS PROPERTY. ✓: AUTHORS SHOW EMPIRICALLY TO WHICH EXTENT THE PROPERTY IS FULFILLED

	white-box							**black-box**																							
Property	[40]	[41]	[36]	[37]	[38]	[39]	[68]	[19]	[18]	[21]	[22]	[23]	[25]	[26]	[27]	[33]	[28]	[29]	[69]	[30]	[43]	[42]	[31]	[32]	[34]	[20]	[70]	[35]	[62]	[61]	[64]
Effectiveness	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fidelity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	N/A	✓
Robustness	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Security	✓	✓	∼	∼	∼		∼	✓	∼		✓		✓	∼	✓	✓								✓				✓	✓		
Legality				∼											✓						∼										
Integrity	✓	✓		✓			✓		✓	✓	✓	✓			✓	✓	✓	✓						✓							
Reliability	✓	✓					∼									✓	✓	✓			✓										
Efficiency	✓	✓	∼		∼	✓	✓							∼		✓			✓		✓		✓								
Capacity	✓		✓		✓	✓					✓					✓												✓	✓		∼

the model's parameters (as in the case of watermarking relational data), this would degrade the model's performance. The model is trained with a regularizer term, given the signature $\mathbf{b} \in \mathbb{R}^T$, the averaged weights vector $\mathbf{w} \in \mathbb{R}^M$, and a specially crafted embedding matrix $\mathbf{M} \in \mathbb{R}^{T \times M}$. The embedding matrix $\mathbf{M}$ can be considered a secret key for the embedding and extracting processes. The watermark is extracted by applying $\mathbf{M} \in \mathbb{R}^{T \times M}$ to the weights vector $\mathbf{w} \in \mathbb{R}^M$ and then applying a step function. The resulting vector $\tilde{b}$ is compared with the signature $\mathbf{b}$, and the bit error rate (BER) is computed. Ownership is proven by thresholding the BER.

Rouhani et al. [40] proposed a watermarking framework, which proves to be more robust against watermark removal, model modifications, and watermark overwriting than [36]. This method is regularizer-based and encodes the signature in the pdf of activation maps obtained at different DNN layers, through an additional regularization term that ensures that selected activations are isolated from others, in order to avoid creating a detectable pattern of alterations. During the verification process, previously generated trigger images are used as input for the model to then analyze the activations. The scheme can be employed in a white- or black-box setting, depending on whether just the output-layer or also hidden-layer activations are assumed to be available for watermark verification. Note that access to the output activations is not guaranteed in a black-box setting.

Wang and Kerschbaum [37] show that both previous schemes are vulnerable to watermark detection (cf. Section IX), as the weight distribution deviated from those of nonwatermarked models. The authors claim that this arises from the additive regularization loss function(s). Consequently, they propose a scheme that is particularly robust against detection attacks. Inspired by the training of GANs, they train a watermarked target DNN, which is competing against a detector DNN that aims to discover whether a watermark is embedded.

Wang et al. [38] followed a similar approach and proposed a white-box scheme that makes use of an additional DNN for the watermark embedding process. The target model is trained in parallel with an embedding model, which is kept secret after embedding. The scheme is regularizer-based, and the watermark is verified by feeding the selected weights into the embedding model and thresholding the output vector. They empirically show that their scheme achieves better fidelity, robustness, and capacity compared to [36].

Feng and Zhang [39] combined a binarization scheme and an accuracy compensation mechanism to reduce the model's accuracy degradation, which is a result of fine-tuning. They use spread-spectrum modulation on the signature $\mathbf{b}$ and embed it in different layers to reduce the risk of the watermarked weights being set to zero during a pruning attack. The binarization scheme then transforms the selected weights per layer so that the second norm of the selected weights in one layer remains unchanged, making it harder to discover the embedding position of the watermark. Finally, they use a regularizer mechanism in fine-tuning to reduce the impact of watermark embedding on the model's performance.

The first (and so far only) white-box framework for automatic speech recognition (ASR), SpecMark, was introduced by Chen et al. [68]. They embedded the watermark in the spread spectrum of the ASR model without retraining it, evaluated SpecMark on the DeepSpeech model, and concluded that it does not have any impact on fidelity.

C. Black-Box Watermarking

Black-box watermarking methods need only querying access to the model during watermark extraction and verification. Only two of the existing black-box watermarking frameworks [42], [43] address the second threat model scenario, i.e., the illegal copy (cf. Section V-A). All the other methods are not reliably robust against model stealing attacks [58] and therefore primarily address the first case (legal copy).

All frameworks that are defending against the legal copy case utilize backdooring via data poisoning (cf., e.g., [72]). A backdoor consists of a so-called trigger set of input–output pairs—which are only known to the backdoor creator (in most cases, the model owner)—and triggers a behavior that is not predictable by others. We call the input images of the trigger set trigger images (sometimes also referred to as watermarks).

Black-box watermarking methods focus on either creating suitable trigger images (inputs) or the output for the trigger image. Depending on the scheme, different trigger images are used for watermarking: out-of-distribution (OOD), pattern-based, noise-based, perturbation-based, and in-distribution. OOD images are completely unrelated to the dataset, for example, abstract images in a dataset of handwritten digits. In-distribution trigger images are taken from the original training dataset and deliberately relabeled wrongly. Pattern-based images are derived from the training dataset, e.g., by marking

Fig. 7. Examples for various types of trigger images, intentionally labeled as a different class [(a) and (b) "cat," (c) and (d) "airplane," and (e) "9"]. (a) OOD [18]. (b) In-distribution [30]. (c) Pattern [19]. (d) Noise [19]. (e) Perturbation [26].

with a pattern such as a logo, text, or other designed patterns. This is comparable to patterns embedded in images for "conventional" data poisoning attacks (see [72]). Noise-based images are derived from the training dataset by adding noise (i.e., no systematic pattern), either visible or invisible to the human eye. Perturbation-based images are slightly perturbed images and lie near the classification boundary; thus, when relabeled, they force the model to slightly shift its classification boundary and are inspired by adversarial examples [9]. Fig. 7 shows the examples for all five types of trigger images. Similar to embedding backdoors—as an attack to reduce the availability or integrity of a model—, the overall objective is that the model will accurately behave on the main classification task while classifying the trigger images as designated by the owner.

Zhang et al. [19] proposed the first black-box watermarking scheme and introduced three types of trigger images: unrelated (OOD), content (pattern), and noise. Their work was the basis for many subsequent papers.

1) Out-of-Distribution: Similar to and shortly after Zhang et al. [19], Adi et al. [18] proposed to include abstract images as triggers in the training dataset. Those abstract images are completely unrelated to the main classification task, and thus, it is highly unlikely that a model that has not seen this data point (i.e., one not watermarked) will label it as the designated class.

One of the first watermarking schemes for image processing models was proposed by Quan et al. [64]. The main difference to classification is that the output is, like the input, an image and not a label—thus, they generate input–output pairs that consist of trigger images and verification images. They use OOD images (or random noise) as trigger images and create the verification images by applying a simple image processing method to the trigger images (ideally not the one on which the model is being trained). The model is then fine-tuned on the union of the original dataset and the trigger set.

Yang et al. [20] empirically showed that distillation is an effective watermark removal attack. Therefore, they propose a scheme that they claim to be especially robust against distillation. The main idea is that the watermark information is carried by the predictions of the original training data, whereas the watermark extraction is done by querying an OOD trigger. In contrast to [18] and [19], the target model is not trained on the union of the original dataset and the trigger set but only on the original dataset while making use of another model, the ingrainer model; this influences the target model by a regularizer term in the loss function. The ingrainer model has the same architecture as the target model and is only trained on the trigger set, with the purpose to overfit the trigger set.

2) Pattern: An improved pattern-based technique was proposed by Li et al. [21]. They showed that previous schemes [18], [19] are vulnerable to ownership piracy attacks, during which an attacker aims to embed their own watermark into an already watermarked model. The authors proposed a scheme that is especially robust against such attacks using so-called dual embedding: the model is trained to classify: 1) data with a predefined binary pattern correctly, i.e., null embedding, and 2) data with an inverted pattern (binary bits are switched) incorrectly, i.e., true embedding. They observe that null embedding does not degrade the model's accuracy if the number of pixels in the pattern is sufficiently small. Furthermore, they evaluated the robustness against model stealing attacks and concluded that with OOD data, the attacker would need significantly more input data to reach similar accuracy.

Guo and Potkonjak [22] proposed to embed a pattern into the trigger images that can be clearly associated with the model owner's signature, e.g., a logo. The pattern should be embedded with little visibility so that an unmarked model would still classify the trigger images according to its original labels.

As an improvement to [22], Guo and Potkonjak [23] proposed an evolutionary algorithm-based method to generate and position trigger patterns. Their algorithm is based on differential evolution [73], an evolutionary algorithm and metaheuristic that searches for solutions to an optimization problem. Using this trigger pattern generation, they demonstrated an improvement in integrity and robustness.

3) Noise: Zhu et al. [25] proposed a watermarking scheme to defend especially against overwriting. They used one-way hash functions to generate both the trigger image and the label. The framework takes an initial image and creates a hash chain of trigger images, as shown in Fig. 8. They showed experimentally that their proposed scheme is robust against overwriting even if the attacker knows the trigger set generation algorithm.

4) Perturbation: The goal of Le Merrer et al. [26] is to slightly shift the decision boundary of the model. This is achieved by generating adversarial examples [9] for images close to the boundary and changing the assigned class label. After fine-tuning the model, the decision boundary is adapted. An illustration of this decision boundary shifting is given in Fig. 9.

Li et al. [27] especially addressed evasion attacks. They proposed a framework closely related to the idea of GANs and used three DNNs: encoder, discriminator, and target model.

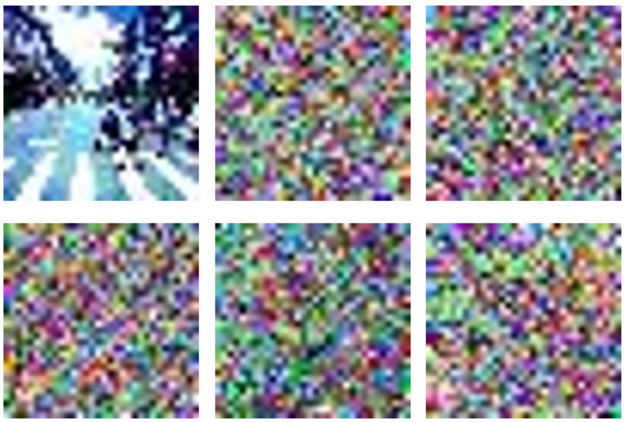

Fig. 8. Top-left image shows the initial image, and the following five are trigger images resulting from a hash chain [25].

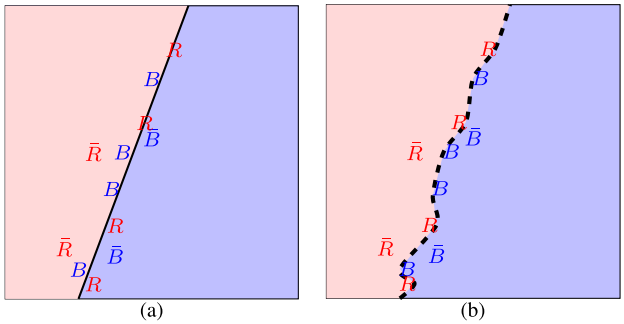

Fig. 9. (a) Data points will be divided into "true adversaries" (R and B) and "false adversaries" ($\bar{R}$ and $\bar{B}$). The label for the true adversaries is changed, and the label for the false adversaries stays unchanged. (b) After fine-tuning, the decision boundary has changed [26].

The encoder takes the original image and aims to embed a logo into the image in a way that the difference is imperceptible. The resulting trigger images are fed into the discriminator—together with the original image—to evaluate the encoder's success. A difference in the original and trigger images is essential for the effectiveness of the embedded watermark—the larger the difference, the better. However, the smaller the difference, the better the protection against evasion attacks—the authors thus specifically aim to address this tradeoff.

5) In-Distribution: Namba and Sakuma [30] proposed an attack called query modification to invalidate the watermark, which exploits the fact that trigger images differ from original training images (cf. Section IX). They then developed a scheme that is more robust, especially against query modification, but also model modifications such as fine-tuning and model compression (e.g., pruning). They suggest to use trigger images that are selected from the training sample distribution. Although the trigger images are undetectable, the model is more likely to overfit the (on purpose) wrongly labeled triggers and, thus, more susceptible to removal attacks via, e.g., pruning. They want to counter this pruning by ensuring that the predictions do not depend on a large number of small model parameters, which would likely be pruned. Therefore, the model is first trained as usual with the original training set. Then, the watermark is embedded through exponentially weighting the parameters and training the model on a combination of the original dataset with the trigger set, which enforces the predictions to depend on a small number of large parameters.

6) Label: The papers in this category focus on the label for triggers. Hence, the choice of trigger images is secondary.

Zhong et al. [31] proposed to label the trigger images with a completely new label rather than assigning an existing one so that the watermark embedding has only little impact on the original classification boundaries. Any pattern-based trigger image can be used in this context. They empirically compare their work to [19] and show that the proposed scheme achieves a zero false-positive rate, i.e., excellent integrity, and is more robust against fine-tuning and model compression.

Zhang et al. [32] observed that trigger images are frequently created in a systematic way, which makes it easier for an attacker to recreate them. Therefore, they propose to include unpredictability in the labels for triggers. They use a chaos-based labeling scheme that ensures that an attacker cannot produce a valid trigger set, even if they know the pattern.

7) Output Vector: The approaches in this category focus on embedding information in the output vector. Chen et al. [33] proposed the watermarking framework BlackMarks, which encodes the signature within the distribution of output activations. To encode the class predictions, the authors design a scheme that maps the class predictions to bits, i.e., by clustering the original classes into two categories represented by bits 0 and bit 1. The trigger images are created as follows: take an image from cluster "0," create an adversarial example so that it would be predicted with a class belonging to cluster "1," and finally label it with a uniformly randomly chosen class from cluster "0." Trigger images for bit 1 are created and vice versa. The watermark is extracted by querying the trigger images and encoding each class to a binary value, which should result in the owner's binary signature.

Similarly, Xu et al. [34] proposed a watermarking scheme that carries the watermark information within the output activations. The trigger pairs consist of a trigger image and a serial number (SN), which is encoded in the model's probabilities.

D. Countering Model Stealing

Only a few schemes address the second threat scenario in Section V-A, i.e., robustness against model stealing attacks [58]. First, Jia et al. [42] proposed a scheme called entangled watermark embedding (EWE). The main idea is to create a watermarked model that is not specialized into "submodels," where one part of the model has learned the main classification task and the other has learned the watermark detection (which is normally lost during a model stealing attack). This is achieved through a regularizer, ensuring that the trigger images lead to similar activation patterns as the original images. Thus, both trigger and original images cause a similar behavior of the model, thereby increasing the robustness against model stealing.

Szyller et al. [43] proposed the framework dynamic adversarial watermarking of neural networks (DAWN), which does not embed a signature into the target model itself, but dynamically returns wrong classes from the API service for a fraction

of queries to mark an adversary model created via a model stealing attack. It is worth noting that the scheme is not able to differentiate between an attacker and a benign client—all clients obtain a fraction of wrong predictions, and it is ensured that the same query always returns the same output (correct or modified) to avoid simple collusion attacks. This approach thus realizes ④ in Fig. 5.

Zhang et al. [61] and Wu et al. [62] proposed, independently of each other, an approach similar to [43], as they are hiding an invisible watermark in the outputs of the image processing model, but for all outputs. When an attacker trains a new (surrogate) model on the input–output pairs of the original model, the watermark will be learned as well and can be verified via black-box access (cf. ④ in Fig. 5). One difference to [43] is that in the case of an image processing model, the output is another image, and thus, there is more space to embed the watermark in. Neither papers explicitly address model stealing attacks, but we believe that they are a suitable defense.

E. Watermarking for Specific ML Settings

Existing watermarking schemes are not suitable for FL, as pointed out by Tekgul et al. [24]. Embedding a watermark in such a setting is different because the model owner has no access to training data, and the training is performed in parallel by several clients. Tekgul et al. [24] proposed to include an independent and trusted third party between the model owner and the clients, which will embed a backdoor-based and black-box watermark into the model at every aggregation step. Furthermore, they propose a specific noise pattern for triggers.

Yu et al. [35] proposed the first watermarking scheme specially crafted for GANs. Previously existing watermarking schemes were limited to DNNs that map from images to classes and thus could not be transferred to GANs. Yu et al. [35] watermarked the input images and then transferred these images to the GAN model. Thereby, the image steganography system, which consists of an encoder and decoder, has first to be trained and, subsequently, all the training data—together with a secret watermark—are fed to the encoder, resulting in watermarked data. The watermarked data are then used to train the GAN model. For verification, the model owner only needs an output image of the GAN and applies the decoder on it to compare the result with the secret watermark. Thus, the proposed scheme needs only black-box access for verification.

F. Watermarking as Part of an IPP Workflow

Besides the abovementioned watermarking techniques as reactive methods, further approaches have been proposed to prove ownership of IP in other domains, e.g., using ledgers such as blockchains to deposit the digital object (or a signature thereof) together with the owners identity [74], [75]. We note that such mechanisms could be one option to prove ownership of an ML model if white-box access to the model exists and the model itself or its signature can thus be compared to the ledger's entries. To be practical, such schemes would need to be robust to small changes in the model parameters—an attacker could perform those changes at little extra loss due to general model robustness, while the changes would invalidate ownership claims, as models and signatures would not match anymore; hence, approaches, such as fuzzy hashing, might be a suitable solution. In addition, as discussed in Section VI, white-box access to a suspected pirated model is unrealistic, as ML models can be exploited and monetized by clients without the need for this type of access.

For black-box methods, which have superseded white-box approaches for reasons outlined in Section VI, access to the model itself is not available, and therefore, no signature can be computed for comparison with a deposited model. To support ownership verification, the aim of the legitimate model owner is to prove that they have knowledge of the trigger set pairs (inputs and expected outputs), which can be seen as a form of challenge–response. As other users of the model could also search for inputs with unexpected outputs (also ex-post with black-box access) and claim these to be valid evidence proving ownership, it is of interest to owners to prove their knowledge at a specific (and earlier) point of time. Depositing signatures of these trigger pairs with a trusted authority (such as a notary) or a distributed ledger (such as blockchain) can provide a trusted time stamp that can be used in the verification process.

G. Length of Watermarks and Complexity

In most cases, the length of the watermark is either determined by the number of parameters changed in white-box watermarking or the number of trigger images used in black-box approaches. Regarding complexity, the embedding time heavily depends on the choice of either training the model from scratch or fine-tuning it, as fine-tuning creates additional overhead. Regarding extraction time, there are no major differences between the methods, as all of them either score trigger images or transform the model's parameters in order to extract the watermark—they are thus all dependent on: 1) watermark's length and 2) the prediction time in case of black-box watermarking or the number of neurons on the chosen layer in white-box watermarking.

VII. Fingerprinting of ML Models

A model owner might be selling their ML model to different customers, but the model gets illegally redistributed by one of them. The owner would then like to gather evidence on the leak; therefore, they could embed fingerprints in the ML model before selling the product in order to trace back a malicious user if needed. We can think of fingerprinting as a user-level extension of watermarking. At the time of performing this systematization, fingerprinting for ML models was not extensively discussed, with only three papers published.

Note that there is another definition of fingerprinting: a (unique) identifier for an object (either hardware, software, or a combination thereof) is generally referred to as a "fingerprint," e.g., such as in browser fingerprinting [76] or device fingerprinting [77]. The application scenario for employing these techniques is often to track devices (resp. their users). Also, this use of fingerprinting is an inherent property of the object and not the result of an active embedding process. Given that this context differs from what we considered so far, we call this form fingerprinting as unique identification.

A. Fingerprinting as User-Specific Watermark

Chen et al. [41] proposed DeepMarks, a white-box fingerprinting framework that is able to embed unique fingerprints. The verification process not only detects the malicious user but also if multiple—and if so which—users collaborated in order to remove the watermark. The embedding process works similar to DeepSigns [40]. The authors propose to assign a unique binary vector (fingerprint) to each user and embed the fingerprint information in the pdf of the weights before distributing the models to the users.

Although DeepMarks is the only paper especially considering fingerprinting, we believe that a couple of the watermarking schemes introduced above can be extended to fingerprinting. To name a few, Uchida et al. [36] embedded a unique signature into the weights of the DNN, Li et al. [27] embedded a unique logo into the trigger images, and Guo and Potkonjak [22] generated unique trigger images based on a signature. All of them could embed user-specific watermarks. Moreover, Xu et al. [34] relied on SNs that can be created in indefinitely many ways, assigning each to a user.

B. Fingerprinting as Unique Model Identifier

Cao et al. [69] proposed a framework to obtain a unique identifier of DNNs. As two different models likely have different classification boundaries, they suggest to "fingerprint" this boundary. The authors identify so-called "fingerprinting data points" that lie near the model's classification boundary. Since the points lie near the classification boundary rather than on it, the authors claim robustness against model modifications and uniqueness of the fingerprint.

Zhao et al. [28] and Lukas et al. [29] modified this idea of fingerprinting as unique identification. Both propose a scheme in which the adversary model—created through applying modifications to the target model—has the same fingerprint as the target model. Both introduced a novel algorithm for creating transferable adversarial examples (see, e.g., [78]). In Section VI-C, we described how black-box watermarking methods use perturbation-based trigger images (i.e., adversarial examples), which are used during training so that the models learn how to (purposefully) misclassify them. In the context of fingerprinting as unique model identifier, the authors want to instead create an adversarial example from an already trained ML model. The key aspect is that these generated images are not only adversarial examples for the target model but also for the adversary model, i.e., they are transferable. This fits our first threat model in Section V-A.

VIII. Access Control and Other Proactive IPP

In this section, we analyze proactive IPP methods. These are orthogonal to and go further than ownership verification.

The most prominent type of methods tries to prevent unauthorized access to a trained neural network. This is achieved by rendering the model useless to an unauthorized user, even if this user manages to obtain a full and exact copy of the model. Most methods employ obfuscation and/or encryption, which can only be overcome with a matching secret.

There are diverging viewpoints on which assets of an ML model are most important to protect. The majority of literature argues that it is the learned model parameters as: 1) learning requires large amounts of (expensive) training data, expertise with training the model, and computing resources and 2) in many cases, standard, well-known architectures (such as GoogLeNet/Inception [79] and ResNet) are employed—the architectures are not secret, but the models need to be (re)trained to fit the domain. However, other works (e.g., [45]) highlight the fact that if a custom architecture is developed, then the resulting structure is actually the asset to protect.

Analogous to ownership verification (cf. Table I), access control mechanisms should primarily fulfill the following requirements.

1) *Fidelity:* The model should maintain accuracy after applying the proactive defense.
2) *Robustness:* The access control should resist a designated class of transformations, including malicious model modifications.
3) *Efficiency:* The impact of the access control mechanism on the time for prediction (and to some extent also for training); this is more relevant than its counterpart for ownership verification (watermarking), as it might affect normal operation efficiency.
4) *Protection Effectiveness:* A model that is used without proper authorization should incur a significant degradation in prediction correctness so that the value for the attacker is diminishing or even nonexistent. For example, Lin et al. [53] set a loss of 20 percentage points in effectiveness as a goal to render the model useless.

The choice of protection scheme also depends on the type of asset to be protected. We can, in general, distinguish the following approaches (cf. Fig. 4):

1) obfuscating the model structure;
2) modifying the input, e.g., by encryption or permutation;
3) encrypting (parts of) the model, i.e., the weights;
4) modifying the model structure, e.g., by adding layers.

We provide an overview of the proactive IPP schemes considered in this article in Table III, where we indicate whether they are meeting the abovementioned requirements. While fidelity and protection effectiveness are discussed or demonstrated by almost all works, we can observe that only a selected number of papers are doing this for robustness against attacks as well as for the efficiency of their scheme.

One aspect common to most access control schemes is that they require the authorized user to possess some form of secret, e.g., a key or a token. Most works, however, do not discuss aspects of management and revocation of these secrets. While those aspects are somewhat orthogonal to the access control mechanism itself, they are of significant importance, as the most commonly discussed scenario is that the models are deployed in the customer's infrastructure, or, e.g., in an embedded device. To some extent, this lack of holistically considering proactive IP protection mechanisms is comparable to using watermarking without considering a proof of existence of the trigger sets (as discussed in Section VI-F).

TABLE III

REQUIREMENTS MET BY ACCESS CONTROL SCHEMES. WE DISTINGUISH TWO DEGREES. ∼: RESPECTIVE AUTHORS CLAIM THAT THE SCHEME FULFILLS THIS PROPERTY. ✓: AUTHORS SHOW EMPIRICALLY TO WHICH EXTENT THE PROPERTY IS FULFILLED

	Structure Obfuscation	**Structure Modification**			**Parameter Encryption & Obfuscation**					**Input Obfuscation**	
Property	[45]	[46]	[47]	[48]	[49]	[50]	[51]	[52]	[53]	[54]	[55]
Fidelity	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	✓
Protection Effectiveness	∼	✓	✓	✓		✓	✓	✓	✓	✓	✓
Robustness		✓	✓				✓	✓			✓
Efficiency	✓					✓	✓		✓		

A. Model Architecture Protection

Xu et al. [45] proposed a scheme to protect the structure of a CNN. They argue that: 1) being able to pirate an architecture, even without trained weights, is a major incentive for adversaries and 2) the most important part of the architecture is the feature extraction in the early layers—and not the fully connected layer(s) at the end. They thus propose to obfuscate these layers of the architecture through iteratively replacing complex processing blocks of a trained CNN, such as an inception module [79], by a small number of sequentially aligned convolutions (the simulation network). For the simpler, shallow structure to learn these modules effectively, they are using teacher–student network approaches, receiving as ground truth the class label and the output of the feature extraction block. The resulting network does not suffer a noticeable effectiveness loss and is in most cases more efficient. The shallow structure is, however, not capable to learn a new, similarly complex task from scratch and, thus, of only limited utility to an attacker.

B. Model Parameter Protection

1) Parameter Encryption and Obfuscation: Gomez et al. [49] proposed a scheme utilizing homomorphic encryption (HE), which allows to compute certain operations directly on encrypted data. Prediction is performed by encrypting the input to the homomorphically encrypted layers and decrypting the resulting output using asymmetric key pairs. Encryption is limited to parts of the model due to the huge runtime overhead incurred by HE. While the authors argue that the last layers they train are the most valuable—and, thus, get encrypted—others (e.g., [46]) argue that the first layers, which are responsible for feature extraction, are the important asset—especially if the training dataset is not public and the feature extractors differ from those of benchmark datasets.

Chakraborty et al. [50] introduced a hardware-protected neural network (HPNN). In their threat model, they assume that an attacker obtains the white-box model and to host their own (public) service or to use it in a private environment. They argue that encrypting the whole NN will lead to prohibitive runtime overhead for predictions, which are, however, often required in (near) real time. They obfuscate the learned parameters through a technique they call locking—i.e., making some neurons in the network dependent on a secret key that determines the sign of the value of the linear function in that neuron. The model can thus be openly distributed, as to correctly use it, a secret key needs to be available in a trusted hardware [root of trust, such as a trusted platform module (TPM)]. The model needs to be trained with a modification of the backpropagation algorithm to be key-dependent, but this does neither affect fidelity nor the model's ability to learn the relationship between inputs and outputs. The authors argue that a hardware solution entails stronger security guarantees and less performance overhead.

Alam et al. [51] proposed using a key-scheduling algorithm to create a series of keys, one for each model parameter of a DNN. After a standard training algorithm, the owner encrypts the parameters using a substitution box (S-box). At prediction time, a legitimate user owning a key uses the same key-scheduling algorithm to create decryption keys for each query.

Lin et al. [53] used chaotic encryption to obfuscate the model parameter positions without changing the weights' distribution, thus making detection of this scheme more difficult. Legitimate users need a key to determine the positions of the output cells of kernel (matrix) operations; otherwise, these outputs will be in a wrong sequence and the prediction correctness will deteriorate. The authors argue that the decryption is fast enough, as only a few layers need to be encrypted, and secure enough, as the decryption can be performed independently for each layer and on the chip, rendering memory attacks impossible. One challenge is selecting the most effective combination of layers to encrypt, as, depending on the architecture, not all layers provide the same protection.

Motivated by SN verification in software products, Tang et al. [52] proposed a scheme for DNNs in which the user has to possess a valid SN. The SN could be a secret combination with the model owner's identity and can therefore be also used for ownership verification. The embedding is done by a teacher–student framework where the teacher network learns the classification task and the student network is distilled from the teacher network, with an additional loss that ensures that the SN is embedded.

2) Input Obfuscation: Chen and Wu [55] proposed a scheme using a transformation module that preprocesses inputs in a secret way before passing them to the prediction module, which needs no further protection—inputs that are not preprocessed correctly will deteriorate accuracy. For the transformation module, the authors invert the idea of adversarial examples: adding an adversarial perturbation specific for each input so that the model correctly classifies them. The module thus acts as a kind of decryption module and is intended to run in a TPM. In their most successful approach, the transformation module is implemented as a CNN and trained together with the prediction module using specific regularizers.

TABLE IV
WHICH ATTACK DEFEATS WHICH WATERMARKING TECHNIQUE BASED ON THE EVALUATION OF THE PAPERS. ∼ DENOTES THAT THE AUTHORS CLAIM THAT THEIR ATTACK CAN BE EXTENDED EASILY TO DEFEAT THIS WATERMARKING TECHNIQUE BUT DID NOT PROVIDE AN EVALUATION FOR THAT

				Watermarking techniques					
	Attack goal	Attack technique	Attack paper	OOD	pattern	noise	perturb.	in-distr.	regulariser
Attacks on watermarks	Invalidation	Substitute ensemble & detector	Hitaj et al. [83]	[18], [19]∼			[26]∼		
		Query pre-processing	Namba and Sakuma [30]	[19]	[19]	[19]	[26]		[40]
	Overwriting	Embed new watermark	Li et al. [21]	[18], [19]	[19]	[19]			
	Detection	Property inference	Shafieinejad et al. [84]	[18], [19]	[19], [22]	[19]			
			Wang and Kerschbaum [37]						[36], [40]
	Detection, removal	Analyse weights variance	Wang and Kerschbaum [85]						[36]
	Removal	Fine-Tuning	Liu et al. [86]	[18], [19]	[19], [22]	[19]			
			Aiken et al. [87]	[18], [19]	[19]	[19]			
			Guo et al. [88]	[18]	[19]		[26]		
			Chen et al. [89]	[18], [19]	[19]		[26]	[30]	
		Distillation	Yang et al. [20]						[36], [40], [41]

Pyone et al. [54] proposed a scheme in which inputs to the model are perturbed in a specific (deterministic) manner through blockwise pixel shuffling before they are fed to the model training or prediction phase. The perturbation is based on a secret that the rightful user possesses—not knowing this key will result in distortion in the spatial arrangement that will render the model ineffective.

3) Structure Modification: Fan et al. [46] proposed passport layers, a scheme of inserting additional layers into the network. These layers are added after the convolutional layers and perform a scaling operation, the parameters of which are derived from the secret, called passport. These are generated, e.g., based on a given set of input images and the values in the feature maps, which result when passing them through a trained model of the same architecture. The author's motivation is not primarily access control, but adding a kind of "second factor" to ownership verification. They argue that it is easy to forge watermarks for a given model and thus have a false ownership claim, e.g., against [36] or [18], as trigger sets based on adversarial examples do not depend on the input data and thus can be obtained from a trained model alone (see Section IX). To demonstrate high fidelity of the model as well is, however, only possible when having authorized access. This additional step for ownership verification could be provided by most of the other access control schemes presented in this section.

Sun et al. [48] showed—using the example of a LeNet-5 CNN—that adapting the activation function of the convolutional layers to be dependent on a random number only known to the legitimate user can provide effective protection.

Lim et al. [47] are the first to propose an access control scheme for a recurrent neural network (RNN). Specifically, they consider an image captioning model producing a text sequence, implemented as a simplified variant of the show, attend, and tell model [80]. The proposed framework is similar to [46], while not embedding the verification information (the owner's key) into the model weights, but into the signs of the hidden states of the RNN. During model inference time, the key is required as input to the model by an elementwise combination with the input data.

C. Unrobust Models as IPP

Szentannai et al. [44] observed that published DNNs are useful as they produce robust predictions even with minor perturbations of the parameters. They thus propose a proactive defense mechanism that renders the model sensitive and fragile through applying transformations that add neurons on any hidden layer of the model. These neurons decompose previously existing neurons in such a way that the mapping between its preceding and subsequent layer is kept, but weights of existing neurons on the modified layer are divided and, thus, more susceptible to small changes in values. As a consequence, even minor modifications of the model parameters, caused by, e.g., fine-tuning, will drastically alter the predictions; subsequently, adversaries cannot utilize the model in a transfer learning setting. In order to make it difficult to spot these additional neurons, the so-called "deceptive neurons," which bear no other functionality, are added as decoys.

IX. ATTACKS ON IPP MECHANISMS

If an attacker knows or suspects that a model is protected, they could try to change the model in order to remove or overwrite the protection. Regarding watermarking, most authors claim that their techniques are robust against various model modifications such as fine-tuning—retraining the model with new data—, and model compression or parameter pruning—setting small parameter values to zero [81], [82]. Still, several attacks that are aiming to remove, overwrite, detect, or invalidate state-of-the-art watermarking schemes have been proposed. We will analyze those in the following.

In Table IV, we summarize the attacks on watermarking schemes. Each line corresponds to an attack and each column to a (type of) watermarking scheme. Table IV shows which attack defeats which kind of watermarking. We list only schemes that were proven to be successfully defeated—missing schemes in the table do not imply strong robustness. We can see that an attack usually addresses either white- or black-box watermarking schemes. The four trigger image types—OOD-, pattern-, noise-, and perturbation-based—seem to be defeated in a similar way. In-distribution watermarks are more difficult to detect or remove, probably because of the fact that they do not differ from the original training data distribution.

A. Watermark Overwriting

Li et al. [21] showed that some schemes [18], [19] are vulnerable to watermark overwriting (they call this "ownership

piracy"). They applied the schemes to four image classification tasks and assumed that an attacker would have access to around 10% of the original training data. They then showed that an attacker could successfully embed its own watermark by fine-tuning the model with data adapted to this watermark.

B. Watermark Detection

Several attacks exploit the fact that a watermarked model actually learns two tasks: the main classification task and the watermark extraction task. Wang and Kerschbaum [85] revealed vulnerabilities against watermark detection when they observed that in regularizer-based watermarking methods such [36], the variance of the distribution of model parameters (they call this weights variance) increases during watermark embedding.

Wang and Kerschbaum [37] showed that regularizer-based watermarking schemes are vulnerable to watermark detection through the use of a property inference attack [90]. Knowing the embedding algorithm, they trained a set of models with similar architecture and similar data (so-called "shadow models"), some of which will be watermarked, others not. From these models, they extract weights as representative features and subsequently train a model on these features to distinguish between watermarked and not-watermarked models. Similarly, Shafieinejad et al. [84] also proposed to use property inference for watermark detection.

C. Watermark Removal

Wang and Kerschbaum [85] further removed watermarks by embedding additional watermarks into the model, following the embedding scheme in [36]. Since every additional watermark might increase the weight's variance, they propose to lower it by adding an L_2 regularizer. Following this procedure, the authors show that the old watermark cannot be extracted, and thus, the model owner cannot claim ownership. It should be noted that although additional watermarks are embedded into the model, the main objective is to "neutralize" the old watermark rather than to use the new watermarks to claim ownership.

Shafieinejad et al. [84] analyzed the robustness of backdoor-based watermarking schemes. In particular, they propose a model stealing attack that trains a substitute model (see [58]). This is performed by querying the original model with a public dataset from the same domain and using the resulting label to train their own model. As the public dataset contains none of the trigger images, the watermark is "lost" in the process. We want to point out that most of the techniques, as per their design, are not robust against model stealing attacks [58], as pointed out by Mosafi et al. [91]. Exceptions, such as EWE [42] and DAWN [43], are described in Section VI-D.

Liu et al. [86] proposed WILD, a framework against backdoor-based watermark techniques embedded via fine-tuning. They argue that it is hard for attackers to collect the required amount of within-domain, unlabeled data for the attack in [84], but that using out-of-domain data impacts the effectiveness of the substitute model too much. Their method requires fewer data, as they augment it by random erasing [92], i.e., removing random segments from the input images. These augmented data alone are, however, not enough to remove a watermark via fine-tuning, due to the high diversity of potential watermarks. The authors note that backdoor patterns are mostly learned by the high-level feature spaces produced by the convolutional layers and not by the fully connected layers. They thus additionally add a regularizer term that ensures a minimal distance in distribution between the high-level feature space of the augmented and the clean dataset during fine-tuning so that a backdoor pattern could not be learned. The authors reveal that it is more difficult to remove OOD, compared to pattern- and noise-based watermarks.

Guo et al.'s removal attack [88] covers two aspects: 1) input data preprocessing consisting of pixel-level alterations such as embedding imperceptible patterns and spatial-level transformation such as affine and elastic transformation, aiming at making the trigger image unrecognizable by the model, and 2) fine-tuning, with data that can be unlabeled and from a different distribution. The second step aims at restoring the accuracy of the model on normal samples, which might suffer from the input data preprocessing. Using the watermarked model as an oracle to obtain labels, these input samples are then preprocessed in the same manner and used for fine-tuning the model. The authors empirically show that their watermark removal attack can remove various types of watermarks without knowledge about the watermark embedding or labeled training samples.

Chen et al. [89][7] proposed REFIT, a watermark removal framework based on fine-tuning. The basis of their work is the phenomenon of catastrophic forgetting [94], which means that models, which are trained on a series of tasks, can easily forget the previously learned tasks. Their attack model assumes that the attacker has no knowledge on neither the watermark nor the watermarking scheme and has limited data for fine-tuning. They first show that in case the training data are known, the watermark can be removed by fine-tuning when choosing the learning rate appropriately. In order to adapt to having only limited data that do not come from the original dataset, the authors include two techniques: 1) elastic weight consolidation (EWC) and 2) augmentation with unlabeled data (AU). EWC slows down the learning of parameters that are important for previously trained tasks, in particular the main classification task, via adding a regularizer term to the loss function. AU, on the other hand, increases the number of in-distribution, labeled fine-tuning data. To this end, unlabeled data are obtained via web scraping and labeled by the pretrained model. In most cases, the model labels the data according to their true classes since the model has not seen the data before, and the watermarked model was trained to fulfill the integrity requirement. The authors showed that the proposed framework successfully removes the watermark from various state-of-the-art watermarking schemes without degrading the test accuracy.

Aiken et al. [87] proposed a method for watermark removal based on previous backdoor removal attacks [95], [96],

[7] Previous version in [93].

assuming an attacker with a small (less than 1%) amount of original training data. Their technique involves three steps. First, they reconstruct the perturbations (backdoor patterns) that are required to flip a sample to the other class, using the method from [95]. Second, they superimpose the pattern on their clean training data to identify neurons that are responsible for recognizing the backdoored images, similar to [96]. The weights incoming to these neurons are then set so that they produce zero activation. Finally, the model is fine-tuned on the clean and backdoored training data while labeling the backdoored training data to the class that is least likely to be watermarked, which prevents reappearance of the neurons that were reset in the previous step. The authors showed that their technique defeats the watermarking schemes [18], [19] by effectively removing neurons or channels in the DNN's layers that contribute to the classification of trigger images.

D. Watermark Invalidation

Watermark invalidation does not aim to remove the watermark but finds a way to render it useless.

Hitaj et al. [83] proposed two such attacks: an ensemble attack and a detector attack. The ensemble attack uses several different models, obtained from, e.g., Model Zoo [56], queries all models, and finally chooses the output that was given by most of the models. If one of the models is watermarked and triggered with a specific input for the watermark extraction process, most likely only the watermarked model will predict the chosen label, while the remaining models will predict the true label. Therefore, the trigger output will not be returned, and the verification fails. The detection attack tries to avoid a trigger response; it trains a neural network, i.e., the detector, which predicts whether the query is intending to trigger a watermark. If the input is recognized as a trigger image, a different or no class at all can be returned. The detector is a binary classifier that needs to distinguish between clean and trigger input. Clean input is collected from other public datasets. Trigger inputs are generated from a portion of these samples. It should be noted that this kind of attack is not able to invalidate pattern-based, noise-based, and in-distribution watermarks, as the detector cannot be trained well for watermark detection without further information about the watermark.

Namba and Sakuma [30] proposed a watermark invalidation attack called query modification processing, consisting of two steps: trigger sample detection and query modification via AE. An AE can reduce the effect of trigger images by diluting the pattern embedded in the original image or by eliminating the embedded noise. Because the application of an AE to nontrigger images impacts the performance of the model on these images negatively, it is not recommended to use the AE on every query. Similar to [83], the authors propose to first detect whether the input could be a trigger image queried during a watermark verification process. They suggest three ways to perform the detection: 1) measuring the effect of the AE on the image in the input space; 2) measuring the effect in the output space; or 3) both. The authors demonstrated to invalidate the watermarks created in [19], [26], and [40].

E. Access Control Invalidation

Besides breaking a (potentially insecure) mechanism underlying encryption or obfuscation, an obvious attack on an access control system is trying to guess a valid secret. The schemes presented in Section VIII all demonstrate that using a wrong secret entails a large drop in fidelity, often to the level of a random classifier. Thus, their vulnerability depends on aspects such as management of the secret or brute-force attacks trying random secrets. The success of these attacks depends on the size or complexity of the secret employed; thus, this can be a decisive factor in selecting a scheme.

Besides these, the most widely studied attack to render access control to ML models ineffective is fine-tuning. Most proposed schemes test for this attack. Xu et al. [45] showed that their scheme is to some extent resistant to fine-tuning, and thus, reusing the pirated network for other tasks is disadvantageous. Chakraborty et al. [50], however, showed that a fine-tuning attack using 10% of the dataset restores the accuracy to 4%–11% of the original accuracy. While this is still potentially large enough to bring little value, it also does not render the network completely unusable. This is addressed by Alam et al. [51], who showed that a model fine-tuning attack—with 10% of the initial number of samples—does not improve the random model accuracy of using a wrong secret. Also, Pyone et al. [54] demonstrated robustness to fine-tuning. Chen and Wu's [55] scheme is vulnerable to a powerful attacker that can observe input–output patterns from the transformation module; depending on their amount, they can then restore prediction accuracy to be within 5%–15% of the original one—which might still be too large to make the attack not worthwhile.

F. Other Attack Considerations

Kupek et al. [97] studied defenses against adversarial attacks. They investigated to what extent secret (defense) parameters—which have an effect on the model parameters, e.g., a weights modification during fine-tuning with an additional loss function—can be estimated by an attacker. If this estimation succeeds, the attack can be tailored to better circumvent the defense. While not primarily studied in the IPP context, this type of parameter estimation could be utilized in attacks against some of the schemes discussed in this article, similar to the vulnerability mentioned by Wang and Kerschbaum [37].

It can be observed that in contrast to reactive methods such as watermarking, there is, at the time of writing, too few works that evaluate proactive schemes—there is especially a lack of works that independently evaluate schemes, i.e., an evaluation done by others than the original authors of the scheme. This might be due to access control techniques being generally newer and, therefore, less explored. However, it indicates a need for a more systematic and thorough theoretical and empirical evaluation of the proposed schemes.

X. Guidelines on Choosing an IPP Method

ML models are certainly an IP that needs to be secured when making it publicly available. Model owners that want to

determine which security measures to take are confronted with a variety of possibilities, which we analyzed and systematized in this article. Based on this work, we can derive a set of guiding questions that will help to decide which action to take:

Do I want to proactively protect my model from malicious users or react in case of a threat event? If proactively, one should consider model access mechanisms or unrobust models (cf. Section VIII); if reactively, watermarking would be an appropriate choice (cf. Section VI).

When needing ownership verification, can I ensure to get full access to the adversary model? If access is not ensured, a black-box approach is the appropriate choice; otherwise, both white box (cf. Section VI-B) and black box (cf. Section VI-C) are suitable, where white-box watermarking schemes tend to have higher fidelity than black-box watermarking.

How am I going to distribute my model? In the case of an API service, one should be aware of model stealing attacks (cf. Section VI-D). Most of the introduced methods are not robust against this type of attack, except EWE [42] and DAWN [43]. When distributing the full model, one should choose an IPP that is robust against model modifications—since that is what an attacker would most likely do before redistributing—or decide whether access control is of importance.

If distributed to multiple users, do I need to be able to trace back a malicious user? If yes, one should consider embedding fingerprints (user-specific watermarks) each time before distributing a model to the users. DeepMarks [41] is so far the only explicitly designed fingerprinting framework that allows unique fingerprint embedding and also detects if users collaborated. However, we believe that other watermarking methods could be extended to fingerprinting, as fingerprinting is a user-specific watermark (cf. Section VII-A).

Is my model large enough to hold additional watermark or model access information? Model owners should be aware that the larger a model, the better it will perform on fidelity since the model has enough "space" for holding the additional information without compromising test accuracy.

Do I already have a trained model? Most watermarking methods and some model access techniques, e.g., [52], [55], embed the information when training the model from scratch. Although it is possible to embed the information later on, fidelity is in this case often compromised. Some of the watermarking methods need an already trained model, e.g., [26] for generating adversarial examples or [30]. If in possession of an already trained model, the model owner can utilize those watermarking methods or implement every other watermarking method but has to be aware of the fidelity loss. Regarding model access techniques, similar observations hold true—some methods, such as [52] and [55], adapt the training process with an additional regularizer to embed information, and thus, most often lead to higher fidelity if they are already employed during training and not during fine-tuning on a previously trained model.

How much effort can I expect an attacker to spend on defeating my security mechanism? Attacks require different amounts of training time, (substitute) training data, different levels of access, and so on (cf. Section IX). This needs to be balanced with the (expected) value which the attack might yield. When choosing an appropriate watermarking method, one should be aware that most methods face a tradeoff between robustness and fidelity.

There are watermarking techniques that have not been broken so far, such as the schemes in [20], [21], [23], [25], [27], [31], [32], [33], [34], [35], [37], [38], [39], [42], [43], [62], [64], [68], and [70] (cf. Tables I and IV). However, it does not follow that these methods are more robust than others, especially as many of these schemes are rather new—and many schemes have only been tested against some attacks.

Regarding model access, we note that most schemes have not been vetted against attacks developed by researchers other than the original authors of the scheme, and more empirical evaluation is required. Thus, at this point, it is not possible to accurately estimate the required effort of an attacker.

XI. Conclusion

IPP for ML assets is a very active research field, but still in its infancy. With a growing number of threats discovered, novel protection methods proposed and counterattacks developed, the lack of a unified view on the vulnerabilities hinders comprehensive approaches. In this article, we performed a systematic review of the field and provide a comprehensive taxonomy of IPP methods for ML models. We further categorized attacks on IPP mechanisms and discussed which specific mechanisms are affected by the attacks. This provides IP holders with a holistic overview of appropriate mechanisms so that they may perform a detailed investigation for a concrete setting.

We note that there is a lack of methods that holistically address multiple threats and attack models. Combining, e.g., model access control systems with other proactive measures such as unrobust models and embedded watermarks, would provide protection against multiple types of attacks. However, there might also be effects that multiple IPP strategies interfere with each other, especially if they need to modify similar aspects of the ML model. With this survey and systematization of knowledge, we provide a starting point in this direction and inform about the complexity of a successful IPP of ML models. Future research is mandated to strengthen and extend evaluation frameworks for IPP methods in ML.

In order to make future work on IPP protection methods and respective attacks more comparable, it is important to establish a benchmark setting with well-defined tasks, evaluation metrics, and artifacts. Evaluating protection and attack methods on a common set of architectures and datasets fosters direct comparison. Thus, reusing previously utilized datasets and architectures is highly encouraged; if this is not possible, newly created artifacts (e.g., not yet employed datasets, model architectures, trained models, and similar artifacts) need to be made available to the research community in an easy and reliable manner, with enough details to understand and reuse them. Also, reproducibility of the experiments is vital in order to ensure that others can compare novel work to previously published results and, especially if new artifacts are used, are able to employ existing methods on these artifacts. Thus, a detailed documentation of the training process and the hyperparameters used is required.

REFERENCES

[1] W. Michiels, "How do you protect your machine learning investment," Tech. Rep., Mar. 2020. [Online]. Available: https://www.eetimes.com/how-do-you-protect-your-machine-learning-investment/

[2] H. Chen, B. D. Rouhani, X. Fan, O. C. Kilinc, and F. Koushanfar, "Performance comparison of contemporary DNN watermarking techniques," 2018, *arXiv:1811.03713*.

[3] F. Boenisch, "A systematic review on model watermarking for neural networks," 2020, *arXiv:2009.12153*.

[4] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Dept. Comput. Sci., Univ. Durham, Durham, U.K., Tech. Rep., EBSE-2007-01, 2007.

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learn. Workshop*, Montréal, QC, Canada, 2014, pp. 1–9.

[6] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27. Red Hook, NY, USA: Curran Associates, Inc., 2014, pp. 1–15.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12, 2019.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[9] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Banff, AB, Canada, Apr. 2014, pp. 1–10.

[10] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 11, pp. 3365–3385, Nov. 2020.

[11] A. B. Kahng et al., "Watermarking techniques for intellectual property protection," in *Proc. 35th Annu. Design Automat. Conf.*, San Francisco, CA, USA, 1998, pp. 776–781.

[12] M. Kamran and M. Farooq, "A comprehensive survey of watermarking relational databases research," 2018, *arXiv:1801.08271*.

[13] X. Zhong, P.-C. Huang, S. Mastorakis, and F. Y. Shih, "An automated and robust image watermarking scheme based on deep neural networks," *IEEE Trans. Multimedia*, vol. 23, pp. 1951–1961, 2020.

[14] S. S. Sharma and V. Chandrasekaran, "A robust hybrid digital watermarking technique against a powerful CNN-based adversarial attack," *Multimedia Tools Appl.*, vol. 79, nos. 43–44, pp. 32769–32790, Nov. 2020.

[15] E. Quiring and K. Rieck, "Adversarial machine learning against digital watermarking," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Rome, Italy, Sep. 2018, pp. 519–523.

[16] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "FPGA fingerprinting techniques for protecting intellectual property," in *Proc. IEEE Custom Integr. Circuits Conf.*, Santa Clara, CA, USA, Aug. 1998, pp. 299–302.

[17] Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting relational databases: Schemes and specialties," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 1, pp. 34–45, Jan. 2005.

[18] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *Proc. USENIX Secur. Symp.* Berkeley, CA, USA: USENIX Association, Aug. 2018, pp. 1615–1631.

[19] J. Zhang et al., "Protecting intellectual property of deep neural networks with watermarking," in *Proc. Asia Conf. Comput. Commun. Secur.*, Incheon, South Korea: ACM Press, May 2018, pp. 159–172.

[20] Z. Yang, H. Dang, and E.-C. Chang, "Effectiveness of distillation attack and countermeasure on neural network watermarking," 2019, *arXiv:1906.06046*.

[21] H. Li, E. Wenger, S. Shan, B. Y. Zhao, and H. Zheng, "Piracy resistant watermarks for deep neural networks," 2019, *arXiv:1910.01226*.

[22] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *Proc. Int. Conf. Comput.-Aided Design*, San Diego, CA, USA, Nov. 2018, pp. 1–8.

[23] J. Guo and M. Potkonjak, "Evolutionary trigger set generation for DNN black-box watermarking," 2019, *arXiv:1906.04411*.

[24] B. G. A. Tekgul, Y. Xia, S. Marchal, and N. Asokan, "WAFFLE: Watermarking in federated learning," in *Proc. 40th Int. Symp. Reliable Distrib. Syst. (SRDS)*, Chicago, IL, USA, Sep. 2021, pp. 310–320.

[25] R. Zhu, X. Zhang, M. Shi, and Z. Tang, "Secure neural network watermarking protocol against forging attack," *EURASIP J. Image Video Process.*, vol. 2020, no. 1, Sep. 2020.

[26] E. Le Merrer, P. Pérez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Comput. Appl.*, vol. 32, no. 13, pp. 9233–9244, Aug. 2019.

[27] Z. Li, C. Hu, Y. Zhang, and S. Guo, "How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, San Juan, PR, USA, Dec. 2019, pp. 126–137.

[28] J. Zhao, Q. Hu, G. Liu, X. Ma, F. Chen, and M. M. Hassan, "AFA: Adversarial fingerprinting authentication for deep neural networks," *Comput. Commun.*, vol. 150, pp. 488–497, Jan. 2020.

[29] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, May 2021, pp. 1–18.

[30] R. Namba and J. Sakuma, "Robust watermarking of neural network with exponential weighting," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Auckland, New Zealand, Jul. 2019, pp. 228–240.

[31] Q. Zhong, L. Y. Zhang, J. Zhang, L. Gao, and Y. Xiang, "Protecting IP of deep neural networks with watermarking: A new label helps," in *Advances in Knowledge Discovery and Data Mining*. Cham, Switzerland: Springer, May 2020, pp. 462–474.

[32] Y.-Q. Zhang, Y.-R. Jia, X. Wang, Q. Niu, and N.-D. Chen, "DeepTrigger: A watermarking scheme of deep learning models based on chaotic automatic data annotation," *IEEE Access*, vol. 8, pp. 213296–213305, 2020.

[33] H. Chen, B. Darvish Rouhani, and F. Koushanfar, "BlackMarks: Blackbox multibit watermarking for deep neural networks," 2019, *arXiv:1904.00344*.

[34] X. Xu, Y. Li, and C. Yuan, "A novel method for identifying the deep neural network model with the serial number," 2019, *arXiv:1911.08053*.

[35] N. Yu, V. Skripniuk, S. Abdelnabi, and M. Fritz, "Artificial fingerprinting for generative models: Rooting deepfake attribution in training data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14448–14457.

[36] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proc. ACM Int. Conf. Multimedia Retr.*, Bucharest, Romania, Jun. 2017, pp. 269–277.

[37] T. Wang and F. Kerschbaum, "RIGA: Covert and robust white-box watermarking of deep neural networks," in *Proc. Web Conf.*, Ljubljana, Slovenia, Apr. 2021, pp. 993–1004.

[38] J. Wang, H. Wu, X. Zhang, and Y. Yao, "Watermarking in deep neural networks via error back-propagation," in *Proc. Int. Symp. Electron. Imag.*, Jan. 2020, pp. 1–9.

[39] L. Feng and X. Zhang, "Watermarking neural network with compensation mechanism," in *Knowledge Science, Engineering and Management*. Cham, Switzerland: Springer, Aug. 2020.

[40] B. Darvish Rouhani, H. Chen, and F. Koushanfar, "DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Renton, WA, USA: Providence, Apr. 2019, pp. 485–497.

[41] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "DeepMarks: A secure fingerprinting framework for digital rights management of deep learning models," in *Proc. Int. Conf. Multimedia Retr.*, Ottawa, ON, Canada, Jun. 2019, pp. 105–113.

[42] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *Proc. 30th USENIX Secur. Symp. (USENIX Secur.)*, Aug. 2021, pp. 1937–1954.

[43] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "DAWN: Dynamic adversarial watermarking of neural networks," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 4417–4425.

[44] K. Szentannai, J. Al-Afandi, and A. Horváth, "Preventing neural network weight stealing via network obfuscation," in *Proc. Sci. Inf. Conf.* Cham, Switzerland: Springer, Jul. 2020, pp. 1–11.

[45] H. Xu, Y. Su, Z. Zhao, Y. Zhou, M. R. Lyu, and I. King, "DeepObfuscation: Securing the structure of convolutional neural networks via knowledge distillation," 2018, *arXiv:1806.10313*.

[46] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, Inc., Dec. 2019, pp. 1–11.

[47] J. H. Lim, C. S. Chan, K. W. Ng, L. Fan, and Q. Yang, "Protect, show, attend and tell: Empowering image captioning models with ownership protection," *Pattern Recognit.*, vol. 122, p. 108285, Feb. 2022.

[48] L. Sun, Y. Wang, and L. Dai, "Convolutional neural network protection method of Lenet-5-like structure," in *Proc. 2nd Int. Conf. Comput. Sci. Artif. Intell.*, Shenzhen, China: ACM Press, Dec. 2018, pp. 77–80.

[49] L. Gomez, M. Wilhelm, J. Márquez, and P. Duverger, "Security for distributed deep neural networks: Towards data confidentiality & intellectual property protection," in *Proc. 16th Int. Joint Conf. e-Bus. Telecommun.*, Prague, Czech Republic: Science and Technology Publications, 2019, pp. 1–9.

[50] A. Chakraborty, A. Mondai, and A. Srivastava, "Hardware-assisted intellectual property protection of deep learning models," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2020, pp. 1–6.

[51] M. Alam, S. Saha, D. Mukhopadhyay, and S. Kundu, "*NN-Lock*: A lightweight authorization to prevent IP threats of deep learning models," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 18, Feb. 2022, Art. no. 3505634.

[52] R. Tang, M. Du, and X. Hu, "Deep serial number: Computational watermarking for DNN intellectual property protection," 2020, *arXiv:2011.08960*.

[53] N. Lin, X. Chen, H. Lu, and X. Li, "Chaotic weights: A novel approach to protect intellectual property of deep neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 7, pp. 1327–1339, Jul. 2021.

[54] A. Pyone, M. Maung, and H. Kiya, "Training DNN model with secret key for model protection," in *Proc. IEEE 9th Global Conf. Consum. Electron. (GCCE)*, Kobe, Japan, Oct. 2020, pp. 818–821.

[55] M. Chen and M. Wu, "Protect your deep neural networks from piracy," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Hong Kong, Dec. 2018, pp. 1–7.

[56] *Model Zoo*. Accessed: Mar. 16, 2022. [Online]. Available: https://modelzoo.co/

[57] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)* Austin, TX, USA: USENIX Association, Aug. 2016, pp. 601–618.

[58] D. Oliynyk, R. Mayer, and A. Rauber, "I know what you trained last summer: A survey on stealing machine learning models and defences," *ACM Comput. Surveys*, pp. 1–36, 2023, doi: 10.1145/3595292.

[59] A. Sablayrolles, M. Douze, C. Schmid, and H. Jegou, "Radioactive data: Tracing through training," in *Proc. Int. Conf. Mach. Learning*, Jul. 2020, pp. 8326–8335.

[60] S. Abdelnabi and M. Fritz, "Adversarial watermarking transformer: Towards tracing text provenance with data hiding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2021, pp. 121–140.

[61] J. Zhang et al., "Model watermarking for image processing networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, Apr. 2020, pp. 12805–12812.

[62] H. Wu, G. Liu, Y. Yao, and X. Zhang, "Watermarking neural networks with watermarked images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 7, pp. 2591–2601, Jul. 2021.

[63] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA: Association for Computing Machinery, Oct. 2017.

[64] Y. Quan, H. Teng, Y. Chen, and H. Ji, "Watermarking deep neural networks in image processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 1852–1865, May 2021.

[65] X. Zhao, H. Wu, and X. Zhang, "Watermarking graph neural networks by random graphs," in *Proc. 9th Int. Symp. Digit. Forensics Secur. (ISDFS)*, Elazig, Turkey, Jun. 2021, pp. 1–6.

[66] V. Behzadan and W. Hsu, "Sequential triggers for watermarking of deep reinforcement learning policies," 2019, *arXiv:1906.01126*.

[67] V. M. Potdar, S. Han, and E. Chang, "A survey of digital image watermarking techniques," in *Proc. 3rd IEEE Int. Conf. Ind. Informat.*, Perth, WA, Australia, 2005, pp. 709–716.

[68] H. Chen, B. Darvish, and F. Koushanfar, "SpecMark: A spectral watermarking framework for IP protection of speech recognition systems," in *Proc. INTERSPEECH*, Oct. 2020, pp. 2312–2316.

[69] X. Cao, J. Jia, and N. Z. Gong, "IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Hong Kong, May 2021, pp. 14–25.

[70] X. Guan, H. Feng, W. Zhang, H. Zhou, J. Zhang, and N. Yu, "Reversible watermarking in deep convolutional neural networks for integrity authentication," in *Proc. 28th ACM Int. Conf. Multimedia*, Seattle, WA, USA, Oct. 2020, pp. 2273–2280.

[71] Y. Nagai, Y. Uchida, S. Sakazawa, and S. Satoh, "Digital watermarking for deep neural networks," *Int. J. Multimedia Inf. Retr.*, vol. 7, no. 1, pp. 3–16, Mar. 2018.

[72] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.

[73] R. Storn, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, pp. 341–359, Dec. 1997.

[74] K. Wust and A. Gervais, "Do you need a blockchain?" in *Proc. Crypto Valley Conf. Blockchain Technol. (CVCBT)*, Jun. 2018, pp. 45–54.

[75] A. Savelyev, "Copyright in the blockchain era: Promises and challenges," *Comput. Law Secur. Rev.*, vol. 34, no. 3, pp. 550–561, 2018.

[76] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*, vol. 6205, Berlin, Germany: Springer, 2010.

[77] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 2, pp. 93–108, Apr. 2005.

[78] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–24.

[79] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9.

[80] K. Xu et al., "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, pp. 2048–2057.

[81] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning," in *Proc. Int. Conf. Learn. Represent.*, San Juan, PR, USA, May 2016, pp. 1–14.

[82] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," in *Proc. Int. Conf. Learn. Represent. Workshop Track*, Vancouver, BC, Canada, Apr. 2018, pp. 1–11.

[83] D. Hitaj, B. Hitaj, and L. V. Mancini, "Evasion attacks against watermarking techniques found in MLaaS systems," in *Proc. 6th Int. Conf. Softw. Defined Syst. (SDS)*, Rome, Italy, Jun. 2019, pp. 55–63.

[84] M. Shafieinejad, N. Lukas, J. Wang, X. Li, and F. Kerschbaum, "On the robustness of backdoor-based watermarking in deep neural networks," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, Jun. 2021, pp. 177–188.

[85] T. Wang and F. Kerschbaum, "Attacks on digital watermarks for deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Brighton, U.K., May 2019, pp. 2622–2626.

[86] X. Liu, F. Li, B. Wen, and Q. Li, "Removing backdoor-based watermarks in neural networks with limited data," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Milan, Italy, Jan. 2021, pp. 10149–10156.

[87] W. Aiken, H. Kim, S. Woo, and J. Ryoo, "Neural network laundering: Removing black-box backdoor watermarks from deep neural networks," *Comput. Secur.*, vol. 106, Jul. 2021, Art. no. 102277.

[88] S. Guo, T. Zhang, H. Qiu, Y. Zeng, T. Xiang, and Y. Liu, "Fine-tuning is not enough: A simple yet effective watermark removal attack for DNN models," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Montreal, QC, Canada, Aug. 2021, pp. 3635–3641.

[89] X. Chen et al., "REFIT: A unified watermark removal framework for deep learning systems with limited data," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Hong Kong, May 2021, pp. 321–335.

[90] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, p. 15.

[91] I. Mosafi, E. O. David, and N. S. Netanyahu, "Stealing knowledge from protected deep neural networks using composite unlabeled data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[92] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, Apr. 2020, pp. 13001–13008.

[93] X. Chen et al., "Leveraging unlabeled data for watermark removal of deep neural networks," in *Proc. ICML Workshop Secur. Privacy Mach. Learn.*, Jun. 2019, pp. 1–6.

[94] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," 2013, *arXiv:1312.6211*.

[95] B. Wang et al., "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 707–723.

[96] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Research in Attacks, Intrusions, and Defenses*. Cham, Switzerland: Springer, 2018.

[97] T. Kupek, C. Pasquini, and R. Böhme, "On the difficulty of hiding keys in neural networks," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, Denver, CO, USA, Jun. 2020, pp. 73–78.

Isabell Lederer received the M.Sc. degree in technical mathematics from the Technical University of Vienna, Vienna, Austria, in 2021. Her master's thesis was based on research on intellectual property protection for machine learning models, in particular watermarking methods for convolutional neural networks.

After her studies, she is pursuing a professional career as a Data Scientist.

Rudolf Mayer is currently a Senior Researcher and the Lead of the Machine Learning and Data Management Team, SBA Research, Vienna, Austria, and a Lecturer with the Vienna University of Technology, Vienna. His research interests include information retrieval (focusing on text and music data) and machine learning. Specifically, he focuses on privacy-preserving data publishing and machine learning, and security aspects of machine learning (adversarial machine learning) and intellectual property protection in machine learning processes.

Andreas Rauber is currently a Professor with the Data Science Research Unit, Department of Information Systems Engineering, Vienna University of Technology, Vienna, Austria; the Head of the Vienna Scientific Cluster Research Center, Vienna; and a Key Researcher at SBA Research, Vienna. His research interests cover the broad scope of data science, ranging from reproducibility and transparency aspects in data analytics and their realization in virtual research environments to explainability and accountability in machine learning.