总的来说，watermark方法需要重新训练模型，fingerprint方法只比对模型

Deep Intellectual Property: A Survey

Yuchen Sun, Tianpeng Liu, Panhe Hu, Qing Liao, Shouling Ji, Nenghai Yu, Deke Guo, Li Liu*

Abstract—With the widespread application in industrial manufacturing and commercial services, well-trained deep neural networks (DNNs) are becoming increasingly valuable and crucial assets due to the tremendous training cost and excellent generalization performance. These trained models can be utilized by users without much expert knowledge benefiting from the emerging "Machine Learning as a Service" (MLaaS) paradigm. However, this paradigm also exposes the expensive models to various potential threats like model stealing and abuse. As an urgent requirement to defend against these threats, Deep Intellectual Property (DeepIP), to protect private training data, painstakingly-tuned hyperparameters, or costly learned model weights, has been the consensus of both industry and academia. To this end, numerous approaches have been proposed to achieve this goal in recent years, especially to prevent or discover model stealing and unauthorized redistribution. Given this period of rapid evolution, the goal of this paper is to provide a comprehensive survey of the recent achievements in this field. More than 190 research contributions are included in this survey, covering many aspects of Deep IP Protection: challenges/threats, invasive solutions (watermarking), non-invasive solutions (fingerprinting), evaluation metrics, and performance. We finish the survey by identifying promising directions for future research.

Index Terms—Intellectual Property Protection, Deep Learning, Watermarking, Trustworthy Artificial Intelligence, Literature Survey

1 Introduction

During the past decade, under the joint driving force of big data and the availability of powerful computing hardware, Deep Neural Networks (DNNs) [1]–[4] have made revolutionary progress, and are now applied in a wide spectrum of fields including computer vision [5], [6] , speech recognition [7], natural language processing [8], autonomous driving [9], cancer detection [10], medicine discovery [11], playing complex games [12]–[14], recommendation systems [15], and robotics [16]. Acknowledgedly, it is of high cost to obtain high-quality models, as training accurate, large DNNs requires the following essential resources:

- *Massive amounts of high-quality data* that are costly to gather and manage, usually requiring the painstaking efforts of experienced human annotators or even experts;
- *Expensive high-performance computing hardwares* like GPUs;
- *Advanced DNNs themselves* that require experienced experts to design their architectures and tune their hyperparameters, which is also costly and time consuming.

As aforementioned, well-trained DNN models have become valuable assets. In addition, recently "Machine Learning as a Service" (MLaaS) has emerged as a popular paradigm to enable many people other than machine learning experts to train DNNs. Therefore, trained DNNs have high business value.

Because of the great value of trained DNNs, many security related issues have arisen [17], [18]. Adversaries can illegally download, steal[1], redistribute, or abuse the trained DNN models[2] without respecting the Intellectual Property (IP) of legitimate parties, leading to serious loss of the model owners. Therefore, to prevent such malicious attacks, there is a pressing need to develop methods to confidentially yet robustly protect the trained models and their IP, enhancing the trustworthiness of DNN models.

Y. Sun (sunyuchen18@nudt.edu.cn) and D. Guo (dekeguo@nudt.edu.cn) are with the College of Systems Engineering, National University of Defense Technology (NUDT), Changsha, China. Li Liu (dreamliu2010@gmail.com), Tianpeng Liu (liutianpeng2004@nudt.edu.cn), and Panhe Hu are with the College of Electronic Science, NUDT, Changsha, Hunan, China. Qing Liao is the the Harbin Institute of Technology (Shenzhen), China. Shouling Ji is with College of Computer Science and Technology at Zhejiang University. Nenghai Yu is with School of Cyber Science and Security, University of Science and Technology of China, Hefei, Anhui, China.
The corresponding author is Li Liu.
This work was partially supported by the National Key Research and Development Program of China No. 2021YFB3100800.

1. create a copy of the DNN model with (almost) identical behavior, also called model extraction

2. including the training data, optimized hyperparameters, learned model weights, *etc.*

The ideal protection for Deep IP is to prevent and alarm model stealing in the first place, or at least degrade its cost performance. However, the leakage risks of DNN models is inevitable. For instance, recent literature [17] proposes a strong tool to defeat most applied methods for terminal-side protection, like encryption, isolation and obfuscation. Literature [18] proposes a guide on model stealing and shows the incompleteness of the current protection lineup. Therefore, the discovery and traceability of leakage models is an essential capability in Deep IP protection, which is mainly focused in this paper.

To this end, an intuitive idea is to endow the target DNN model with a model-specific identifier that can be extracted by model owners or institutions for IP Protection with a correct secret key. Once the leaked model or its prediction are online, the model leakage will be discovered through the identifier, so as to promote subsequent behaviors of rights protection and responsibility investigation. According to the way of identifier registration, it could be categorized into two types:

模型需要修改

- The first are invasive solutions named as watermarking, where *'invasive'* refers that the target model has to be modified during identifier registration. DNN watermarking refers to embed the identifier (*i.e.*, a <secret key, watermark message>pair) into model parameters, gradients, structures, or outputs, through finetuning or retraining the target model. Note that the secret keys could be matrices or trigger samples.
- The other are non-invasive solutions named as fingerprinting with no model modification. The intuition is that the piracy version has a similar decision boundary to the victim model, while individually trained models are quite different. Thus, fingerprinting extracts a unique model properties as the identifier to measure model similarities, reflect by near-boundary test cases and designed test metrics like model predictions or neuron activation.

借助模型的特征

The early works focused on model watermarking techniques, and have made some progress in robustness, security and other issues. However, such solutions require model modifications, resulting in potential security risks and additional burdens. Recently, some works have explored model-based fingerprinting techniques. So far, although Deep IP protection has received wide attention and many outstanding works have been proposed, the existing schemes still cannot meet the demands of various applications and the threat of endless attack modes. Deep IP protection is still

但watermark也有优点，比如支持blackbox

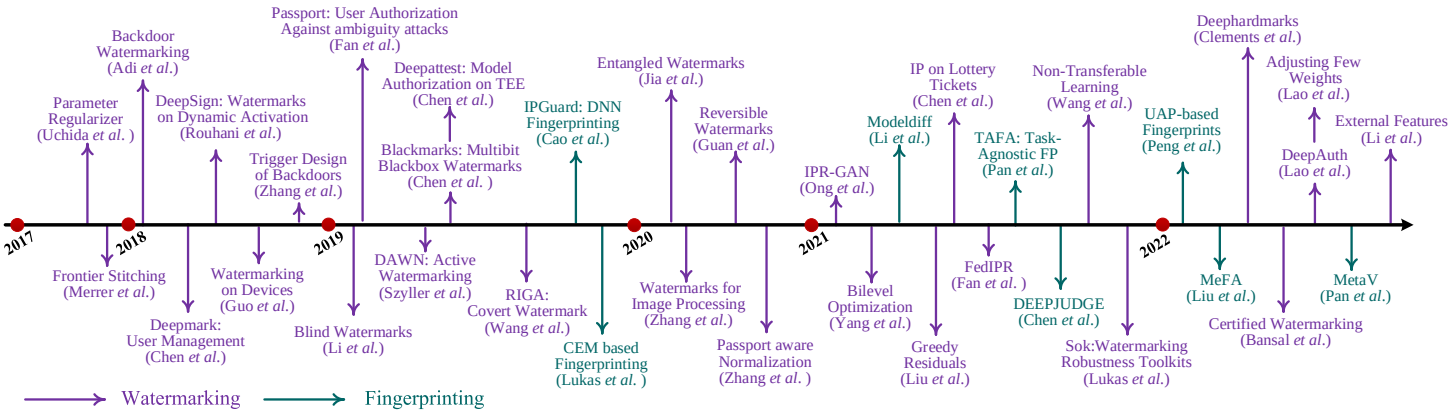


Fig. 1. Chronological overview of representative methods for Deep IP Protection from the first contribution in 2017 to the latest, including model watermarking and fingerprinting. Watermarking is are to embed watermark messages by modifying target models. Some of them embed a bit string with a regularizer on selected parameters [19]–[21] or normalization layer [22], [23]. The others finetune target models by trigger samples with predefined labels [24]–[27]. In contrast, fingerprinting-based methods are to depict the decision boundary by constructing near-boundary samples and test metrics [28]–[35]. Most of these methods compare the similarity between model with a single test metric [28]–[33]. Several methods design multiple test metrics for stronger robustness [34], [35]. See Section 5 and 6 for details.

in its early stages, which motivates us to systematically review recent progresses of this field, identify the main challenges and open problem impeding its development, to clarify the promising future directions. On account of the diversity of potential Deep IP solutions, there is still a lack of systematic investigation. To fill this gap, we present this survey for a comprehensive overview.

1.1 The Scope of this Survey

Embedding watermarks into deep neural networks

Figure 1 provides a chronological overview on the evolution of the young field over the past five years since the pioneering work conducted by Uchida *et al* [19] in 2017 on DNN watermarking. This survey summarizes all of the research works that we are aware of during this period, including DNN watermarking and fingerprinting techniques, and presents a multi-view taxonomy, main challenges, methods, evaluation criteria, and performance, together with current shortcomings and potential valuable research orientations. DNN watermarking and fingerprinting are involved in this survey together in three reasons. First, these two techniques follow the same *registration-verification* framework, and accomplish the same goal to detect whether the model is illegally stolen, distributed or tampered. Fingerprinting could be regard as an improvement of watermarking to avoid modifying trained models. Second, they have similar basic techniques, evaluation metrics, and challenges, which reflects the current inadequacies of the field. Third, these two types of techniques basically cover all methods to detect the attacks on Deep IP on the model side, which will help to understand the field of Deep IP more comprehensively. We hope that this survey will provide readers with a clear picture of similarities and differences between existing approaches, as well as an taxonomy to reveal some open problems in the field through the proposed.

互补关系

1.2 Differences with Related Surveys

Although several surveys on Deep IP have been published [18], [36]–[42], however, to the best of our knowledge, there is no survey to provide a comprehensive review covering DNN watermarking and fingerprinting techniques. We present a brief review of related surveys and emphasize our new contributions.

Some surveys [40], [42] only focus on DNN watermarking. For example, [43] discusses the dissimilarities between media and DNN watermarking and identifies some challenges for future research. Ffkirin *et al* [42] review some watermarking methods and test the robustness against the fine-tuning attack using several optimizers on MNIST and CIFAR10 datasets. Li *et al* [40] present a novel taxonomy to divide DNN watermarking methods into static and dynamic categories. Only a small part of watermarking methods are included in them. Some surveys [36], [37] discuss fingerprinting techniques; however, they either lack a clear definition or the investigated research works are not comprehensive enough to date. Regazzoni *et al* [36] only review 13 research works. Boenisch *et al* [39] propose a definition of fingerprinting similar to ours, but only two works based on fingerprinting are reviewed. Xue *et al* [37] proposed a taxonomy on Deep IP Protection from 6 views: scenario, mechanism, capacity, type, function and target models, and divided the attack methods into three groups: model modifications, evasion/removal attacks, and active attacks. Most works identified as fingerprinting in literature [37] are put into the watermarking category in ours. [44] reviews some typical Deep IP Protection methods, which can be divided into passive protection and active protection from the view of protection purpose, and into static and dynamic from the view of technical implementation. In contrast, we focus on passive protection and classify watermarking and fingerprinting methods more fine-grained. From the perspective of testing protocol, Chen *et al* [41] propose an experimental performance comparison on fidelity, robustness and integrity. Lukas *et al* [38] propose Watermarking-Robust-Toolkit (WRT) to test the robustness of some model watermarking methods. [45] evaluates the robustness of some trigger-based methods. However, they only compare some classical watermarking methods in the early stage, neglecting other metrics and advanced methods. Compared with these surveys, we propose the most comprehensive review to date including more than xxx research contributions and many aspects of Deep IP Protection. From the threats to Deep IP, Oliynyk *et al* [18] present a comprehensive review on model stealing and its defence from a high-level perspective. However, they mainly focus on stealing methods, and Deep IP Protection is only briefly mentioned. Literature [46], [47] only reviews model attacks, like stealing and poisoning, without any glance at Deep IP Protection. These works show the serious threats to Deep IP and reflect the urgency of this survey from the attack side.

Our major contributions can be summarized as follows:

- To the best of our knowledge, this is the first systematic and comprehensive survey covering DNN watermarking and fingerprinting techniques for Deep Intellectual Property.
- We present the problem definition, evaluation criteria, testing

protocol, main challenges and threats of Deep IP Protection.
- A multi-view taxonomy is proposed to categorize existing methods and reveal the open problems. Moreover, we analyze the performance of representative approaches on the proposed evaluation criteria, as well as their underlying connections.
- We discuss the current limitations and give some suggestions for prospective research directions.

Protecting Deep Intellectual Property
- **Invasive Solutions: Watermarking (Section 5)**
  - **Inner-component-embedded Watermarking (Section 5.2)**
    - *Static Model Weights: Raw or Transformed*
    - *Dynamic Hidden-layer Activations or Gradients*
    - *Model Structures for Watermarks or Watermarking Structures*
    - *Extra Components*
  - **Trigger-injected Watermarking (Section 5.3)**
    - *Generating Trigger Samples*: simply processed or optimized
    - *Labeling Trigger Samples*: Hard or Soft Predictions
    - *Embedding Paradigms*
  - **Output-embedded Watermarking (Section 5.4)**
  - **Combined Usage of DNN Watermarking**
  - **Watermarking for Specific Applications (Section 5.5)**
    - *Self-supervised learning*
    - *Transfer learning*
    - *Federated learning*
    - *Other AI Applications*
- **Non-Invasive Solutions: Fingerprinting (Section 6)**
  - **Comparing Model Weights (Section 6.2)**
    - *Training Path Recovery*
    - *Weight Hash: Random Projections or Learnable Hash*
  - **Comparing Model Behaviors (Section 6.3)**
    - *Model & Data Preparation (Section 6.3.1)*
    - *Test Case Generation (Section 6.3.2)*
      - Samples with simple or even no processing
      - Optimized Near-Boundary Samples
    - *Test Metric Designs (Section 6.3.3)*
      - Simple Soft/Hard Predictions
      - Specially-Designed Test Metrics
    - *Fingerprint Comparison (Section 6.3.4)*
      - Simple Similarity Thresholds
      - Hypothesis Testing
      - Learnable Meta-Classifiers
    - *Target Scenarios, Models, & Functions (Section 6.3.5)*

Fig. 2. A taxonomy of representative model watermarking and fingerprinting methods for Deep IP Protection.

Paper Organization. The remainder of this paper is organized as follows: The problem, motivation, functions and evaluation criteria are summarized in Section 2. In Section 3, we present the framework of Deep IP protection from the vies of Deep IP registration and verification. In Section 3, we introduce the challenges and threats to Deep IPs. In Section 5 and 6, we present a taxonomy of the existing methods for Deep IP protection including the invasive watermarking methods and the non-invasive fingerprinting methods, as shown in Fig. 2. Then in Section 7, we provide an overall discussion of their performance (Tables 5). Followed by Section 8 we conclude the important future research outlook.

2 Background

2.1 The Problem & Motivations

A deep neural network (DNN) is a learnable function $\mathcal{M}: \mathcal{X} \rightarrow \mathcal{Y}$ that represents the transformation from a probability measure $\mathcal{X} \in \mathbb{R}^d$ to the target $\mathcal{Y} \in \mathbb{R}^k$. $\mathcal{M}$ is a composition of functions $g_L \circ f_L \circ g_{L-1} \circ f_{L-1} \cdots \circ g_1 \circ f_1$ including linear functions $\{f_i | i = 1...L\}$ and non-linear activation functions $\{g_i | i = 1...L\}$. Generally, each linear function f_i has two types of learnable parameters, weights $\boldsymbol{\omega}_i$ and biases $\boldsymbol{b}_i$, and the full parameter

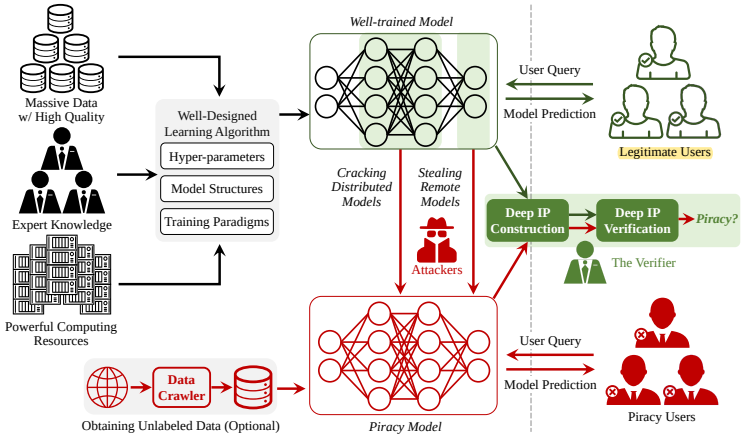


Fig. 3. The Problem and Motivations of Deep IP Protection. A well-trained DNN model requires a heavy drain on multiple resources including data, hardware, and algorithm designs. The MLaaS paradigm allows legitimate users without strong training capability to access well-trained models through model distribution or remote APIs of service providers. However, malicious users can also create a piracy version through stealing remote models or cracking distributed models. Therefore, Deep IP Protection, to register an IP signature of the target model, is in urgent needs for piracy detection.

set of a DNN is denoted as $\Theta = \{\boldsymbol{\omega}_i, \boldsymbol{b}_i | i = 1...L\}$. To train a DNN $\mathcal{M}$, a differentiable loss function $\mathcal{L}$ should be determined to measure the approximation error between the DNN's output $\mathcal{M}(\mathcal{X}; \Theta)$ and the target $\mathcal{Y}$. Then, the loss function $\mathcal{L}$ should be minimized by gradient descent on the learnable parameters Θ, as

$$\min_{\Theta} \mathcal{L}\left(\mathcal{M}(\mathcal{X}; \Theta), \mathcal{Y}\right) \tag{1}$$

Benefiting from the powerful fitting ability to continuous transformations, DNNs propose unprecedented performance in many tasks, such as image classification, object detection and natural language processing, and show great commercial value in a wide range of industry applications including medical, finance, advertising, etc. However, overparameterized DNN models imply high costs for model design and training to achieve strong generalization performance. As shown in Fig. 3, a well-trained DNN model requires three essential supports as follows:

(i) *Massive Data with High Quality:* The primary factor for the success of deep learning is the development of massive high-quality data, such as ImageNet [48], JFT-300M [49], MS-COCO [50], Google Landmark [51], and many more high-value commercial datasets that are not open source. The collection and labeling of datasets costs expensive manpower and time resources, especially in the fields that require scarce domain experts and acquisition equipment, like remote sensing and medicine.

(ii)*Precious Expert Knowledge:* Excellent deep learning models and algorithms require long-term efforts from academia and industry experts. This includes careful designs of model structures, training strategies, hyperparameters, task execution pipelines, and other aspects. Close collaboration between professionals is essential, from deep learning techniques to target task domains, from research exploration to engineering techniques to project management.

(iii)*Powerful Computing Resources:* In general, a larger model trained on a larger training dataset will have better performance. Meanwhile, it also requires more computing resources including not only strong hardware such as GPUs, but also high-performance distributed architectures.

Any of the aforementioned prerequisite supports requires a lot of investment. However, high-cost well-trained DNN models are faced with serious risks of theft. DNN models provide

the following two paradigms. The first requires delivering model parameters and structures, such as pre-trained models opened to downstream tasks and efficient models packaged into mobile apps. Although around 60% of model providers protect their models by licenses or encryption techniques to users, the models are still at high risk of being leaked and attackers are motivated and able to crack the distributed models [17]. The other widespread paradigm is Machine Learning-as-a-Service (MLaaS), that is, opening remote APIs of complex DNN models for clients via a pay-per-query principle, where the models are deployed at the cloud servers of service providers and the clients upload the query data to the cloud then obtain the returned model prediction. However, this paradigm is faced with the risk of model extraction attacks. Adversaries can create a piracy version of the target model through some *query-prediction* pairs while the existing defence strategies are insufficient for endless attack mechanisms [18].

The leakage of DNN models will not only damage the commercial revenue of model owners, but also bring many unpredictable consequences once criminals steal and utilize the DNN models illegally. For example, criminals can maliciously leverage generative models (e.g., GAN, DDPM) to generate fake faces and videos, or leverage the model vulnerability to avoid the Person Re-identification of intelligent monitoring devices. Considering the incompleteness of current defence techniques against model stealing, it becomes crucial to research the post-hoc protection strategies for DNNs' Intellectual Property (Deep IP).

Post-hoc Deep IP Protection refers that the model owners or law enforcement agencies can discover or even track the pirated copies of their own DNN models after adversaries' infringement acts, and provide legally-effective evidence to protect the interests of legitimate owners. And beyond that, as one of the key techniques for Trustworthy Artificial Intelligence (TAI), Post-hoc Deep IP Protection can be extended to various fields, such as model tamper-proofing, user authentication and management, and the traceability of model distribution chains. The workflow of Deep IP protection mainly consists of two stages: Deep IP Registration and Deep IP Verification: i) The former embeds or constructs an identifiable and model-distinguishable **IP signature** of the target model to reflect the copyright information of owners, like a bit string, a specific pattern, or the labels of predefined trigger samples. It is usually achieved by model watermarking or fingerprinting. ii) The latter is to extract the IP signature given a suspect model, then compare the extracted IP signature with the registered IP signature of the protected model. A piracy model will retain an IP signature similar to that of the protected model, but an individually trained model will not.

2.2 Functions

The area of Deep IPs has considered the following basic functions:

- *Copyright Verification (CopyVer):* It means to verify the ownership of a given model by comparing the IP signatures between the given model and the owner's model. Most current works focus on this situation. In general, the IP signatures for copyright verification should be robust to model modification, and irreversible for a strong unforgeability and non-removability.
- *Integrity Verification (InteVer):* It means to detect whether the protected model is modified illegally during model distribution or runtime execution. In contrast, InteVer is often accomplished through fragile or reversible IP signatures. The 'fragile' means that the IP signatures would be changed if the protected model is perturbed slightly, such as the inversion of several bits or finetuning at a small learning rate. The 'reversible' means that the protected model is able to return to the original model using a correct secret key, however, the modified version cannot.
- *User Management (UM):* It means to verify the authorized user of a given model. An intuitive application is to discover the source of the piracy model, *i.e.*, which user leaked it from, so as to facilitate subsequent accountability.
- *Access Control (AC):* It means to prevent unauthorized users from accessing the protected model or limit the model functionality (e.g., the applicability of data domains [52]). Only the users with correct secret keys can utilize the model for model inference, or finetune the model to adapt downstream tasks through transfer learning. Note that this survey does not focus on parameter the access control schemes based on parameter encryption or obfuscated [53]–[55].

Beyond DNN models: Data Protection. Deep IPs also require to protect valuable data from illegal model training, which can be implemented in two ways or their combination: i) the first is to make the models trained on unauthorized datasets with detectable IP signatures belonging to the data owners; ii) or we can make the models trained on unauthorized datasets unusable, that is, significantly damaging the model accuracy. The former generally adopts the watermarking or fingerprinting methods [56], so we classify these methods according to the taxonomy proposed in this survey as shown in Fig. 2. Note that the latter is not the focus of this survey [57].

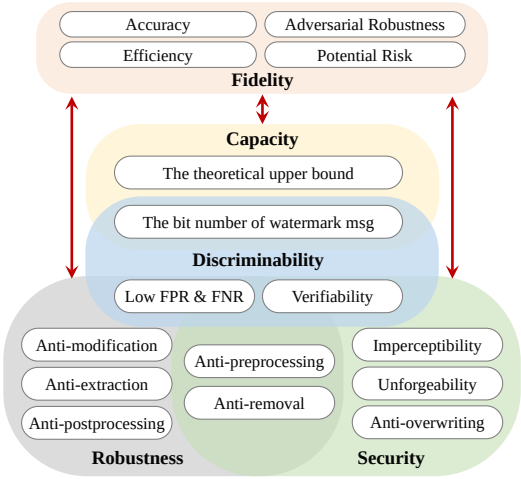


Fig. 4. Illustration for the requirements and the evaluation criteria of Deep IP Protection. An ideal protection method should consider both fidelity and discriminability well. Fidelity means that the protected model should have close performance with the raw model on accuracy, efficiency and robustness. The fingerprinting solution without any model modification has the best fidelity. Discriminability means that, the protected model has enough information and assurance to claim the copyright, though malicious attackers strive to break the registered IP signatures or the verification process. It is reflected by three sub-metrics: robustness, security and capacity. See details in Table 1.

2.3 Requirements & Evaluation Criteria

An ideal method for Deep IP Protection should satisfy the two main requirements as follow meanwhile:

- *Fidelity:* IP signatures could be registered by modifying the target model, such as finetuning model parameters or adding some extra components to the model. The basic requirement is to ensure that the protected model and the raw model have close Quality of Services (QoS). Fidelity is the metric to measure the difference of models' QoS, including accuracy, efficiency, adversarial robustness and potential risks. The *"potential risk"* means that, a modified model may produce unexpected results on some inputs, or it may be easier for a malicious attacker to corrupt model predictions through backdoors.

TABLE 1
The Metrics of Deep IP Protection with Description

Metric	Description
Fidelity	Compared with the target model before IP registered, the protected model should complete the same tasks with similar performance, such as model accuracy, inference efficiency, adversarial robustness, and potential risks.
Discriminability	The extracted IP signature should have sufficient information and assurance to claim the intellectual property, including i) extremely low false positive rate (FPR) for individually trained models and false negative rate (FNR) for protected models; ii) verifiability that ensures the uniqueness of IP signatures; iii) enough bit number to represent the copyright information of owners.
Robustness	Even if the secret keys and host signals (*i.e.,* model parameters or predictions) are perturbed, the IP signatures should be correctly extracted. It primarily consists of two aspects: i) The IP signatures of the protected model should remain integrally in the piracy models, that is, the modified or extracted version of the protected model via fine-tuning, pruning, compression, distillation, or transfer learning. ii) For remote verification, the trigger samples or their returned predictions could be perturbed by piracy model providers, such as compression or noising on inputs and perturbation or smoothing on outputs.
Capacity	The payload of Deep IP signatures is an important criterion for concealing adequate information about model copyright or user identification for Deep IP Protection. Moreover, an IP signature is generally harder to crack or forge if it has a higher capacity, which could be likened to the fact that passwords or hash codes with more bits are harder to be cracked under the same mechanism. The meaning of capacity contains two aspects: the theoretical upper bound, and the bit number of the actual payload in IP signatures.
Security	Users without full authorization (i.e., verification algorithms and secret keys) should be prohibited to detect, extract, re-embedded, modify, forge, and delete the IP signatures of protected models. This requires that the protection strategy can resist as many active attack means as possible, including but not limited to the following three points: i) *Non-trivial proof.* An adversary cannot construct a key pair $(\mathcal{K}; \boldsymbol{b})$ in advance to claim the ownership of an arbitrary unknown model, even if he knows the protection algorithm. ii) *Unforgeability.* Even if the adversary knows the registered IP signatures $\boldsymbol{b}$, he cannot convince a third party that he has the model ownership without knowing the secret key $\mathcal{K}$; iii) *Unremovability.* An adversary cannot remove the registered IP signatures even if he knows its existence and the protection algorithms.

- *Discriminability:* It means that model owners or verifiers should have enough information and assurance to claim the copyright of protected models. It has the following aspects: i) By comparing the IP signatures of models, verifiers can determine whether a given model is a copy of the protected model. This verification process should have low false positive rates (FPR) for individually trained models and false negative rates (FNR) for protected models. ii) Enough capacity, *i.e.,* the bit number of the IP signatures, is necessary to show the powerful evidence of infringement. iii) More importantly, a trained model usually needs to be modified when applied to downstream tasks, *i.e.,* , fine-tuned on the target data domain. Alternatively, when a malicious attacker creates a pirated copy, the attacker generally considers removing the possible IP signatures in various ways such as model modification and extraction, or corrupting the process of IP Verification through input/output processing and ensemble attacks. The ideal method for Deep IP protection should be able to work well in this challenging environment.

Note that the two requirements need to be balanced in practical usage. In theory, fingerprinting methods without any model modification have optimized fidelity, however, their discriminability is generally worse than watermarking methods.

The meaning of discriminability contains many aspects. Here, we extract three main metrics to reflect discriminability: robustness, capacity, and security. Note that these three metrics cannot cover the full meaning of discriminability. Thus, we present these four metrics side by side. Table 1 shows the five metrics for Deep IP Protection along with the description to reflect the two requirements, and it will not be repeated here.

3 The Framework of Deep IP Protection

Fig. 5 illustrates the pipeline of Deep IP Protection consisting of two main stages: IP Registration and Verification. The stage of IP Registration embeds or extracts an identifiable and model-distinguishable signature $\boldsymbol{b}$ for the source model $\mathcal{M}$, e.g. a bit string, to reflect the copyright information of owners. The signature $\boldsymbol{b}$ should be retained by the model owner or a trusted third-party verifier. The owner can extract the signature $\boldsymbol{b}$ from $\mathcal{M}$ when and only when a correct key $\mathcal{K}$ is input. Given a suspect model $\mathcal{M}'$, the verification stage extracts the signature $\boldsymbol{b}'$ and compares $\boldsymbol{b}'$ with $\boldsymbol{b}$. If the difference between the two signatures $dist(\boldsymbol{b}', \boldsymbol{b})$ is below a threshold, the suspect model $\mathcal{M}'$ will be determined as a pirated copy of $\mathcal{M}$.

3.1 Deep IP Registration

According to whether IP Registration needs to modify the protected model, Deep IP Protection schemes can be categorized into two types: DNN watermarking and fingerprinting:

i) *DNN Watermarking* is an invasive solution that embeds at least a detectable identifier (*i.e.,* a watermark message $\mathcal{W}$) into the host DNN model $\mathcal{M}$, to obtain a watermarked model $\mathcal{M}_{\mathcal{W}}$. Generally, DNN Watermarking contains two main components: the Secret Key Generator $\mathcal{G}_{\mathcal{W}}$ and the Watermark Embedder $\mathcal{E}_{\mathcal{W}}$. The Secret Key Generator inputs a host model $\mathcal{M}$ and a candidate dataset $\mathcal{X}_{\mathcal{W}}$ and obtains a secret key $\mathcal{K}$ for watermarking, which is denoted as $\mathcal{K} = \mathcal{G}_{\mathcal{W}}(\mathcal{M}, \mathcal{X}_{\mathcal{W}})$. Then, the generated key-message pair $\{\mathcal{K}, \mathcal{W}\}$ would be input to the Watermarking Embedder $\mathcal{E}_{\mathcal{W}}$ and embedded into the model $\mathcal{M}$ through model fine-tuning or retraining on a dataset $\mathcal{X}$, as

$$\mathcal{M}_{\mathcal{W}} = \mathcal{E}_{\mathcal{W}}(\mathcal{M}, \mathcal{W}, \mathcal{K}, \mathcal{X}) \tag{2}$$

ii) *DNN Fingerprinting* is a non-invasive solution that extracts the unique characteristics called *Model Fingerprints* $\mathcal{F}$ of the source DNN model $\mathcal{M}$ to depict the model similarity. DNN Fingerprinting is mainly based on the assumption that, DNN models trained independently will be significantly different in at least one of the aspects including initial weights, training algorithms or training dataset, and these differences will be reflected in some model features like model prediction, decision boundary, robustness, neural activation paths, etc. On the contrary, since the piracy model is a modification or approximation variant of the victim model, it will also be highly similar to the victim model in model features. Once a suspect model $\hat{\mathcal{M}}$ has a similar fingerprint of the victim model $\mathcal{M}$, *i.e.,* $sim(\mathcal{F}(\mathcal{M}), \mathcal{F}(\hat{\mathcal{M}})) > \epsilon$, we have high confidence to say that $\hat{\mathcal{M}}$ is a piracy copy of $\mathcal{M}$.

As shown in Fig. 5, the process of Fingerprint Registration consists of the Test Case Generator and Test Metric Design. The former selects or generates a set of test cases as the secret key $\mathcal{K} = \{k_i | i = 1...n\}$, and the latter design a series of model features $\mathcal{C} = \{c_i | i = 1...m\}$ that can maximize the fingerprint similarities between the source model $\mathcal{M}$ and its modification or approximation variant $\mathcal{M} + \Delta$ while minimizing the similarity between $\mathcal{M}$ and the independently-trained model $\mathcal{M}_s$. Formally, Fingerprint Registration can be denoted as

$$\begin{aligned} \min_{\mathcal{K},\mathcal{C}} \; & \sum\nolimits_{\mathcal{M}_s} sim(\mathcal{F}_{\mathcal{K},\mathcal{C}}(\mathcal{M}), \mathcal{F}_{\mathcal{K},\mathcal{C}}(\mathcal{M}_s)) \\ & - \sum\nolimits_{\Delta} sim(\mathcal{F}_{\mathcal{K},\mathcal{C}}(\mathcal{M}), \mathcal{F}_{\mathcal{K},\mathcal{C}}(\mathcal{M} + \Delta)) \end{aligned} \tag{3}$$

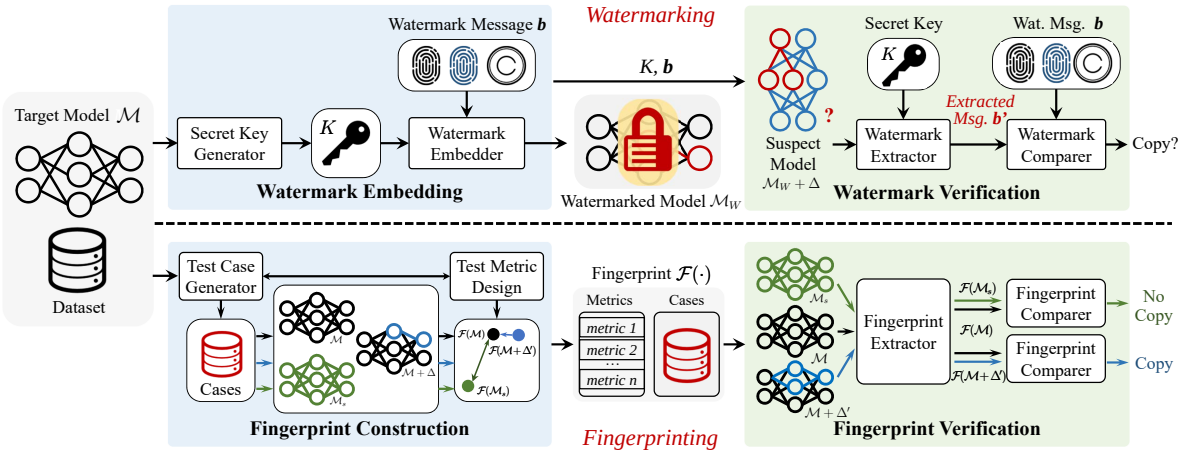


Fig. 5. The Pipeline of Deep IP Protection. The main workflow consists of two stages: Deep IP Registration and Deep IP Verification. From the perspective of IP Registration, the methods for Deep IP Protection could be divided into two groups: Watermarking and Fingerprinting. Watermarking is an invasive solution to embed an IP signature to the target model, generally through model finetuning with trigger samples or a regularizer. In contrast, fingerprinting, to depict the model characteristic with test cases and test metrics, is a non-invasive solution without any model modification. The core idea is that, the piracy version has a similar characteristic to the target model while individually-trained models are significantly different.

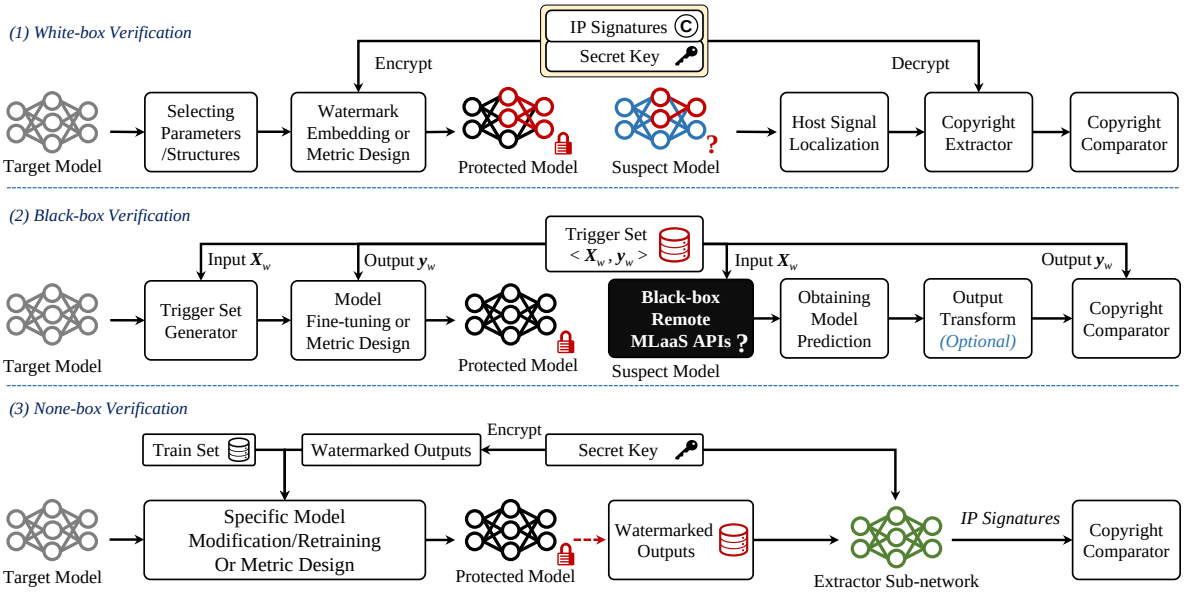


Fig. 6. The Illustration of Deep IP Verification. We group all the verification methods into three categories: white-box, black-box and none-box verification. White-box verification refers that, verifiers have full knowledge of the target models including parameters, structures, etc, and extract the IP signatures through three steps: host signal localization. copyright extraction and copyright comparison. In the scenario of black-box verification, verifiers can only obtain the model prediction of predefined trigger samples such as soft probability or even hard labels; thus, verifiers should extract the IP signatures only from the prediction. Sometimes, verifiers have none of the access permission to the target models, neither model parameters nor inference APIs. However, the output results of piracy models during services could be circulated on the Internet. In this case, verifiers can detect infringement or abusement if the output results contain IP signatures in some forms like image or text watermarks, where a DNN is generally used as the extractor.

In general, test cases $\mathcal{K}$ are distributed near the decision boundary of the protected model. The final key-message pairs $\{\mathcal{K}, \mathcal{F}(\mathcal{M})\}$ are retained by the model owners for IP Verification.

Differences and relations between model fingerprinting and watermarking. It is described from the following perspectives:

- *Purpose*: Both of them aim to construct unique and detectable IP signatures to protect the target models, and realize similar IP functions like copyright or integrity verification by comparing the IP signatures extracted from the suspect models with those of the protected models.
- *Implementation of IP Registration*: Whether watermarks or fingerprints, IP signatures are represented either by properties of internal model components (e.g., the distribution of model parameters) or by model behaviors on test samples (e.g., model outputs). However, unlike watermarking, which modifies or ever retrains the target model to create specific properties or behaviors as IP signatures, fingerprinting is to analyze and extract the intrinsic and discriminating characteristics of the target model without any model modification.
- *Implementation of IP Verification*: The two ways follow similar verification pipelines, *i.e.*, extracting IP signatures from suspect models to be compared with registered IP signatures. There are three ways to compare signatures: threshold comparison, Hypothesis Testing or Learnable Meta-Classifiers.
- *Performance*: They have similar challenges and evaluation metrics. Generally, the discriminability of watermarks is higher than that of fingerprints, but watermarks will damage the fidelity of the target model.
- *Basic Techniques*: Some basic techniques for deep learning, such as DNN interpretability, adversarial attacks and robustness, and probability statistics, have been shared across fingerprinting and watermarking. Watermarking can draw on the techniques like DNN backdoor attacks and communication theories.

3.2 Deep IP Verification

It consists of two main components: Extractor and Comparer. Given a suspected model $\hat{\mathcal{M}}$, the verifier inputs the secret key $\mathcal{K}$ and then extracts the key-message pair $\{\mathcal{K}, \boldsymbol{b}'\}$ through the verification algorithm $\mathcal{V}$ corresponding to the IP Registration method $\mathcal{E}_{\mathcal{W}}$, as

$$\boldsymbol{b}' = \mathcal{V}(\hat{\mathcal{M}}, \mathcal{K}, \mathcal{E}_{\mathcal{W}})$$

where $\mathcal{E}_{\mathcal{W}}$ is used to locate the copyright signature from a DNN model. Then, through a comparer, the verifier determines whether the suspected model $\hat{\mathcal{M}}$ is a true pirated copy of the source model $\mathcal{M}$ according to the similarity between the registered message $\boldsymbol{b}$ and the extracted message $\boldsymbol{b}'$. The methods for IP Verification have different requirements on access permission of the suspected model $\hat{\mathcal{M}}$, and most of them apply one or a combination of the following three verification scenarios:

i) *White-Box Verification*: The verifier requires full access to the suspected models including model parameters and structures in this verification scenario. The first step is to retrieve or recover the host signal, that is, a subset of raw/transformed model weights, model structures, dynamic outputs of some hidden layers, the bias and scales of normalization layers, or various designed metrics in fingerprinting-based schemes. Giving a secret key $\mathcal{K}$ and the location of host signals, the verification function $\mathcal{V}$ will return a message $\boldsymbol{b}'$. Note that the function $\mathcal{V}$ could be a combination of several simple operations or a complex DNN model. The extracted message $\boldsymbol{b}'$ would be similar to $\boldsymbol{b}$, when and only when performing IP verification on the source model $\mathcal{M}$ or its pirated version $\mathcal{M} + \Delta$ with the correct key and signal location.

ii) *Black-Box Verification*: Here, the verifier only requires the predictions of the suspected model, such as soft probability or hard labels of classification/recognition models, the predicted value of regressive models, or the generated data of generative models. It enables remote verification on MLaaS APIs, where the service providers only open the APIs access to DNN models rather than the whole model. After receiving the input samples from users, the service providers perform model inference on remote servers and return the inference result. The verifier uploads the trigger samples $\mathcal{X}_T$ as secret keys $\mathcal{K}$, and obtains a series of input-output pairs $\{\mathcal{X}_T, \mathcal{Y}'_T\}$ to depict the model behaviors as IP signatures $\boldsymbol{b}'$. Some works need to perform some further steps on $\{\mathcal{X}_T, \mathcal{Y}'_T\}$ to extract $\boldsymbol{b}'$. Although deploying MLaaS APIs on remote servers can protect the commercial benefits of DNN models, infringement, *i.e.,* the MLaaS API with a piracy model, becomes more difficult to be detected. First, the verifier can only judge infringement using the model predictions on trigger samples. Second, the procedure of model inference is completely controlled by the service provider. This provides infringers with more means to attack the stage of IP verification besides modifying/retraining target models, like input detection/processing and ensemble attacks. In addition, the verification cost is not neglectable due to the commonly-paid MLaaS APIs.

iii) *None-Box Verification*: Unlike Black-Box verification that requires pre-defined trigger samples, in None-Box Verification, the verifier can extract the copyright message from the model prediction taking normal samples as inputs. Therefore, this scenario enables remote IP verification without explicit secret keys and even any model access permission, which is because the verifier can utilize the generated data spread through the Web during the service of MLaaS APIs. In a way, none-box verification can be regarded as an extension application of traditional data IP protection in DNN models, and it could be also designed by introducing the research of traditional data steganography. The verification function $\mathcal{V}$ is often performed by an extractor DNN model jointly trained with the source model. In general, this verification scenario is applied in generative models whose prediction has enough capacity to hide effective copyright messages. The typical applications include various tasks of image processing and natural language processing, such as semantic segmentation [58], image super-resolution [59], style transfer [60], machine translation [61], etc.

4 Main Challenges & Threats to Deep IP

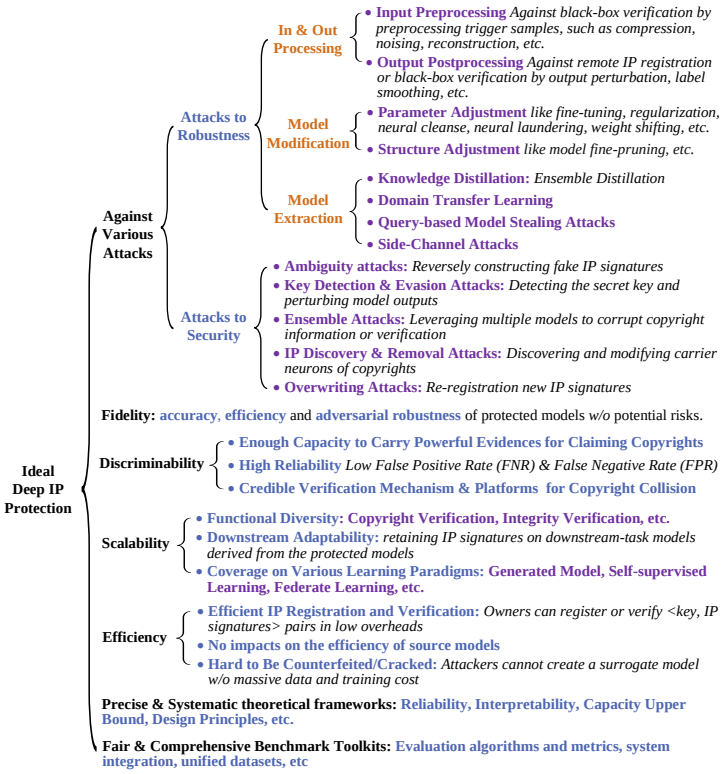


Fig. 7. The Challenges of Deep IP Protection.

4.1 Main Challenges

In open environments of practical applications, Deep IP Protection needs to complete the whole workflow from IP construction, piracy discovery, IP verification to giving effectual evidence, without affecting the quality of service. However, existing protection methods are far from perfect to solve this challenging problem, and we summarize the challenges as follows:

4.1.1 Faced with Various Attacks

In an open environment, Deep IP Protection is facing endless attack methods. As described in Section 3, the workflow of Deep IP Protection includes several steps: IP Registration, IP Verification, and Evidence Demonstration. Attackers can remove (or even overwrite) the registered IP signatures, return an incorrect verification result if a verifier posts a request for Deep IP Verification, and claim ownership of the protected model through a forged IP signature. Once any step is cracked by a malicious attacker, the protection method will be invalid. See details in Section 4.2.

4.1.2 Fidelity

As described in Section 2.3, the protected model should have a similar performance to the target model on accuracy, efficiency and adversarial robustness without potential risks. The watermarking method is better than the fingerprinting methods on discriminability, but worse on functionality-preserving.

4.1.3 Discriminability

As described in Section 2.3, Deep IP Protection should meet the following demands: i) The IP signs should have enough capacity to carry powerful evidence for claiming copyrights; ii) The verification results should have high reliability, *i.e.,* low false positive rate (FNR) and false negative rate (FPR); iii) The protection methods should work even if the protected models are modified legally or maliciously. Additionally, in order to make Deep IP Protection provide a legal effect on resolving copyright conflicts, credible verification mechanisms and platforms are in urgent need.

4.1.4 Scalability and Efficiency

Scalability indicates that the protection methods can be employed in a variety of scenarios, including the following respects: i) *Functional Diversity:* Deep IP Protection should support multiple functions, including copyright verification, user authorization/management, integrity verification, etc; ii) *Downstream Adaptability:* A well-trained model will be applied in various downstream tasks by legitimate users through model finetuning or transfer learning. Therefore, the registered IP signs should be effective in downstream models derived from the protected model; iii) *Coverage on Various Learning Paradigms:* Beyond classification models, various learning tasks and paradigms, such as generated models (eg., VAE, GAN and DDPM), self-supervised learning (eg., MAE), incremental learning, active learning, and so on, have been proposed to encourage a broader range of applications. Not only the well-learned parameters but also how to learn a good model in all of these paradigms should be protected well.

Efficiency means two aspects: First, Deep IP Protection should be highly efficient for both model users and model protectors (owners). In other words, model protectors can register and verify pairs of secret keys and IP signatures with minimal overheads, and model users should be unaware of the impacts on the inference efficiency of protected models. On the other side, it should make malicious attackers as inefficient as possible. It is preferable if attackers are unable to crack protected models or counterfeit IP signatures. Or at least, protectors should raise the cost on data and computation for cracking protected models while degrading the performance of the piracy models, so that the attacker has a weak incentive to crack the protected models.

4.1.5 Lack of Precise & Systematic Theoretical Framework

A precise and systematic theoretical framework is of great importance towards reliable and credible Deep IP Protection. However, unlike traditional IP Protection such as image watermarking, the black-box properties of DNNs make a theoretical examination of Deep IP Protection problematic. The theoretical analysis of Deep IP is still blank at present. Jointly considering the previous works on traditional IP protection and DNN interpretability, this theoretical framework should contain three aspects: i) *The interpretability and explainability of verification results:* The results of IP Verification should be credible and convincing, which requires an interpretable way. In other words, once a verified result determines that a model is a pirated copy of a protected model, it should tell the basis of the judgment as well as the uncertainty. Or the verified result can reflect the degree and direction of decision boundary change between the verified model and the protected model. ii) *The design principles of protection methods to improve functionality-preserving and discriminability:* Existing works have proposed various techniques to improve the performance of Deep IP Protection. However, guiding design principles are still in lack. Facing a specific scenario, an appropriate tradeoff between functionality-preserving and discriminability should be achieved through certain design principles, such as the choices of hyperparameters. iii) *The Upper Bound of Capacity:* Capacity is a crucial goal in the field of traditional digital watermarking and steganography techniques. The same is true for Deep IP. Existing works have explored multi-bit DNN watermarking, however, not focused on the theoretical upper bound of capacity.

4.1.6 Lack of Fair & Comprehensive Benchmark Toolkits

Deep IP Protection is an active field of research, with many research works. A benchmark toolkit for Deep IP is quite essential to guide IP Protectors in selecting optimized methods for different application requirements. However, it is challenging to fairly and comprehensively compare the performance of existing protection methods and their defence capability against diverse attack means. Most of the existing works were evaluated based on their self-designed testing protocols, and only tested a few metrics such as robustness and accuracy. Lukas *et al* [38] proposed an open-source implementation of some classical watermarking schemes and removal attacks and evaluated their robustness. In the future, a more complete testing framework is required, including the testing process covering comprehensive metrics, the construction of unified and representative datasets, and the integration of attack and defence strategies.

4.2 Threats of Deep IP

This section provides an overview of Deep IP threats from the standpoint of attackers' capability on data and models along with the attacked procedures of protection workflows. In Table 2, we summarize some existing methods to attack Deep IPs along with their categories and attributes.

4.2.1 Data-level Abilities of Attackers

We divide attackers' data capability into different levels from the perspective of data quantity, data distribution, and labels; then, describe the purpose of attackers in stealing the model. The first is the data capability of attackers: 攻击者具备的数据假设

- *LLI:* A Large number of Labeled In-distributed data.
- *LLO:* A Large number of Labeled Out-of-distribution data.
- *TLI:* A Tiny amount of Labeled In-distribution data.
- *TLO:* A Tiny amount of Labeled Out-of-distribution data.
- *LUI:* A Large number of Unlabeled In-distributed data.
- *LUO:* A Large number of Unlabeled Out-of-distribution data.
- *TUI:* A Tiny amount of Unlabeled In-distributed data.
- *TUO:* A Tiny amount of Unlabeled Out-of-distribution data.

Note that "*In-distributed*" means that the attacker's data follows the same distribution as the data training the protected model or the data domain of the target task, whereas "*Out-of-distributed*" means that the attacker's data is dissimilar to either of the two so that the model trained on the attacker's data cannot be transferred to the target domain.

Strong attackers (LLI) have enough data to train a good model (of the same order of magnitude as the protected model's train

TABLE 2
Threats to Deep IP Protection

攻击方法

Types	Methods	Data Demands	Model Demands	Literature & Description
IP Detection & Evasion	Detecting White-box Watermarks	≥ Poor	White-box Attackers	Detecting the abnormal distribution of model parameters [62], [63] Detecting the abnormal distribution of hidden-layer activations [64] Detect non-essential or potentially-watermarked components in models
	Detecting Black-box Verification	≥ Poor	White-box Attackers	Detecting Trigger Inputs: Detector [64], GAN [65], JSD [66] Detecting Trigger Channels or Neurons: Neural Laundering [67]
	Input/Output Processing	≥ Poor	White-box Attackers	AE-based Query Modification [66], Input Reconstruction [68], Input Quantization [69], Input Smoothing [70], Input Noising [71], Input Flipping [38], PST [72], Feature Squeezing [70], JPEG Compression [73], Ensemble attack [64], AdvNP [74]
IP Removal	Model Modification	≥ Poor	White-box Attackers	Fine-Tuning (RTLL, RTAL, FTLL, FTAL) [19], [75], Reversely Fine-tuning [65], PST-based Fine-tuning [72] REFIT [76], WILD [77], [78], Weight Pruning [79], Fine-Pruning [80], Label Smoothing [81], Weight Regularization [82],FLAME [83], Weight Shifting [38], Feature Permutation [38], ReMoS [84], Neural Cleanse [85], Neural Laundering [67], Weight Quantization [86], Adversarial Training [87], Attention Distraction [88], Minimal Modification [89], Referenced Subspace Attention [90], Neural Structural Obfuscation against white-box watermarking [91]
		Strong	White-box Attackers	Adaptive Model Modification [45]
	Model Extraction	≥ Medium	Black-box & White-box	Retraining w/ hard labels [82], Smooth Retraining [38] Cross-Architecture Retraining [38], Adversarial Training from scratch [87] Knockoff Nets [92], Transfer Learning [93], Distillation [94], Ensemble Distillation [95]
		Strong	Black-box Attackers	Adaptive Model Extraction [45]
IP Ambiguity	Reverse Construction	≥ Poor	White-box Attackers	Searching a secret key of a given IP signature [22], internal attacks [22]
	Re-registration	≥ Poor	White-box Attackers	Overwriting a new key-signature pair [62]

攻击者对模型的访问等级

set), and their motivations for stealing the model could be: i) discovering the weaknesses of others' models to implement DNN attacks such as adversarial attacks and backdoor attacks; ii) avoiding the computationally heavy process with hyperparameter search for training DNN models; iii) combining others' models to improve the performance of their own models; iv) using ensemble attacks to crack the registered Deep IP signatures of other protected models. In addition, since existing Deep IP schemes still rely on algorithm concealment, model protection mechanisms can be easily defeated if the algorithm is leaked to the attacker in some ways, such as bribing insiders, and mining the vulnerabilities of management mechanisms or software platforms. We consider such attackers as strong attackers.

Medium attackers (LUI) have a sufficient amount of data, but, the quality and labels are insufficient; thus these attackers require the inference results of the protected model to train a DNN model. Note that the black-box attackers should have LLI-level or LUI-level data to execute IP attacks, and attackers with other data levels can only attack white-box models.

Weak attackers (TLI, TUI) do not possess enough data for individual model training or extraction, and they can only crack the registered Deep IP signatures of white-box models through attack methods like model finetuning. TLI-level or TUI-level data allow for some model finetuning while also ensuring model accuracy. By subdivision, TLI's attack capability is slightly greater than TUI's, because TLI does not need to resort to the output of the protected model, thereby degrading the resemblance between piracy models and the protected model.

Poor attackers (LUO, LLO, TUO, TLO) have data that is not sufficiently correlated with the target task. Thus, not only is model extraction prohibited, but also model finetuning should be strictly limited. By subdivision, LUO and LLO may examine model characteristics based on large amounts of OOD data to guide the Deep IP attack process.

4.2.2 Model-Level Abilities of Attackers

Next, we grade attackers' model-level abilities into four levels (level-0 represents the strongest attackers):

- *Level-0:* Attackers have full access to multiple models trained on similar-domain datasets, including model parameters, structure, intermediate feature map, gradient and other information.
- *Level-1:* Attackers have full access to only one protected model.
- *Level-2:* Soft predictions and some intermediate feature maps can be obtained.
- *Level-3:* Only soft predictions can be obtained.
- *Level-4:* Only hard labels can be obtained.

Note that only attackers in the *"level-0"* and *"level-1"* can directly detect, remove, or forge the IP signatures of the protected model, which we refer to as **white-box attackers**. In contrast, attackers in the *"level-1"*, *"level-2"*, and *"level-3"* need to first extract the protected model through stealing attacks like knowledge distillation, named as **black-box attackers**. Attackers in the *"level-0"* are capable of performing ensemble attacks, such as ensemble distillation and ensemble evasion.

4.2.3 The Categorization of Threats

According to the attack procedure in the protection workflow, the threats of Deep IP can be categorized into three types: IP Detection & Evasion, IP Removal, and IP Ambiguity.

- **IP Detection & Evasion: Attacking the verification process**. The verifiers give a secret key for IP and then obtain IP signatures. In the white-box verification scenarios, the verification process is controlled by the verifiers and thus could be considered secure. In this case, IP detection is generally used to aid in subsequent IP Removal attempts, by detecting the specific parameter distributions introduced by watermark embedding. In the black-box scenarios, the verifiers upload trigger samples to the suspected MLaaS APIs and receive the prediction results. This process is controlled by service providers. Therefore, service providers using piracy models will try to attack the verification process, through input/output processing, secret key detection then output perturbation, ensemble attacks, etc. In general, it has two perspectives, *i.e.*, invalidating the input trigger samples, or erasing the hidden IP signature of the prediction returned to the verifier.

检测是否嵌入IP

- **IP Removal: Attacking the registered IP Signatures**. Malicious attackers can destroy the registered IP signatures and create a copied model without effective IP signatures. It could be accomplished through model modification or model extraction. Model modification is to adjust the model parameters or structures, such as model finetuning, fine-pruning, regularization, neural cleanse, etc. The attackers must have full access to the protected model. Model extraction is to learn a piracy model from the prediction behaviors or exact model properties of protected models, such as knowledge distillation (query-based stealing attacks) and side-channel attacks [18]. It can be accomplished in both the black-box and white-box scenarios. Generally, model extraction has stronger performance than the model modification on attacking IP signatures but requires a larger dataset. 模糊IP，使得置信度下降
- **IP Ambiguity: Degrading the confidence of verification results**. After obtaining IP signatures, the verifier needs to provide valid evidence to claim ownership of the model. However, given a protected model, the attackers can re-embed or reversely construct the IP signatures belonging to the attackers, where the former requires finetuning the model and the latter does not. For example, in the black-box verification scenarios, the attackers can construct the trigger sample through adversarial attacks, or fine-tune the protected model with a trigger set predefined by the attackers. Moreover, in the white-box scenarios, the attackers can reversely construct a secret key given a predefined IP sign (a bit string), or re-embed (even overwrite) a predefined IP sign given a secret key.

4.2.4 IP Detection and Evasion

First, the verification process of Deep IPs can be detected or evaded by adversaries. We overview the workflow of this type of attack methods in the following three categories:

- *Detecting White-box Watermarks:* In general, embedding watermarks into the models will significantly change some attributes of the models' internal components. For example, the probability distribution or the frequency spectrum of the watermarked model parameters would be different to that of normal parameters. The hidden-layer activation may also produce a similar phenomenon. Even the extra components as watermarks cannot ensure concealment. Therefore, adversaries can leverage these abnormal attributes to discover the watermarks, like detecting the abnormal distribution of model parameters or hidden-layer activations, or detecting the non-essential or potentially-watermarked components in models.
- *Detecting Remote Black-box Verification:* Remote Black-box Verification is often realized by uploading some trigger samples to suspected MLaaS APIs and then analyzing the return results. The MLaaS providers can detect whether an input sample is a trigger sample, and then perform some specific operations, like denying services or label modification, to circumvent remote IP verification. Trigger sample detection can be done by analyzing the positions (e.g., likelihood) of the input samples in the distribution of task samples. Generally, watermarking schemes select out-of-distribution (OOD) or near-OOD samples as trigger samples in order to easily change the predicted labels with minor model modifications, while fingerprinting schemes select the adversarial samples close to the decision boundary for analyzing the model similarity. However, these operations will significantly change the distribution of samples, which can be detected by the verifiers but also exposed to the adversaries. Moreover, the neurons activated by the trigger samples would also be significantly different from the normal task samples, which can be leveraged by adversaries to avoid IP verification.
- *Input Preprocessing and Output Postprocessing:* Since the verifiers have no knowledge about the details of the internal operation process of the suspected MLaaS APIs, the MLaaS providers can modify the input samples or the inference results arbitrarily. Besides traditional lossy data compression techniques, the adversaries can use adversarial or backdoor defence techniques at the sample level to evade remote IP verification. Moreover, if an adversary has multiple models with different IP signatures, the adversary can combine the inference results of multiple models to obtain an inference result without any IP signature. A simple voting method can implement this process.

4.2.5 IP Removal

Voting可以视为一种混淆IP检验的方法，reusing？

As shown in Table 2, a large number of methods have been proposed to remove the IP signatures of IP-protected models by model modification or extraction.

- *Model Modification:* It refers to removing the IP signature of the target model by modifying the model parameters or structures. This type of IP removal attacks can be implemented using a few data samples. Besides the untargeted methods, like model fine-tuning, pruning, and their variants, some IP-aware modification methods have been proposed. For example, adversaries can try to find and then modify the parameters or neurons that are sensitive to IP signatures but unrelated to tasks. Moreover, if an adversary has the algorithmic information about the adopted Deep IP scheme, the adversary can simulate the trigger samples by generative models or preset rules and give these generated samples the correct labels to fine-tune the target model.
- *Model Extraction:* If an adversary has a certain number of in-distribution data samples, the adversary can obtain the inference results of the protected models on these samples, and then use these results as the soft labels to retrain a piracy model from re-initialized model parameters. Model extraction can effectively remove IP signatures and maintain model performance to a certain extent. Firstly, inner-component-embedded watermarks are generally vulnerable to model extraction since model extraction does not depend on model internals and can be even done across model architectures. Then, specific output patterns on trigger samples are also difficult to transfer to extracted models, since trigger samples are usually weakly coupled to task samples. In the same way as model modification, active attacks can also be performed in model extraction if the adversary has the algorithmic information about the adopted Deep IP scheme. Moreover, if an adversary has multiple models for similar tasks, the adversary can leverage the inference results from these models to learn an ensemble model without any IP signature, by which the model accuracy could be even improved.

4.2.6 IP Ambiguity

In addition to IP detection, evasion and removal, adversaries can also fabricate copyright conflicts, such as creating their own IP signatures by optimizing the secret keys, or re-registering their own IP signatures of the target models.

- *Reverse Construction:* Given a target model, an adversary can set its own forged IP signatures and adjust the secret keys to make the verification step output the forged IP signatures. In this way, the paired secret key and IP signature are constructed by the adversary, thereby confusing the IP signatures registered by the model owner to create copyright conflicts. This attack can be accomplished even though the target model is not allowed

to be modified. For example, the watermarking methods based on model parameters are generally is generally done by fine-tuning the raw or transformed model parameters using a regularizer loss. However, fine-tuning the key matrix using the same regularizer loss has a similar effect. It can be done by a projected gradient descent on the key matrix. For trigger-based watermarking, the techniques for adversarial sample generation can also be used to forge trigger samples with specific labels.

- *IP Re-registration:* An adversary can use its own deep IP methods to construct an IP signature on the target model. In this way, IP signatures belonging to the owner and the adversary will have copyright conflicts.

Remark 1: For some Deep IP schemes against IP Ambiguity, even if it is difficult to re-register IP signatures using the same algorithms, However, adversaries can use other Deep IP algorithms or design their own Deep IP algorithms to complete IP Ambiguity attacks. In this way, if there is no unified standard in the industry, that is, adopting a unified algorithm to achieve Deep IP protection, the current IP Ambiguity attacks are still effective and difficult to defend. However, since the current Deep IP schemes still rely on the concealment of algorithms, it is still not feasible to form a unified industry standard that can be published.
Remark 2: When dealing with copyright conflicts, since model thieves usually do not have the raw model, *i.e.*, the clean model without any IP signature, the judicial organization can require the party to provide the raw model to prove the ownership. However, for fingerprinting schemes, the raw model is not modified. Even for watermarking schemes, due to the transferability of adversarial samples, the clean model may output the prediction results containing forged IP signatures on the trigger samples provided by the adversary. Moreover, an adversary may have multiple model sources for similar tasks, and may fool the judiciary via the ensemble version of these models, even if the adversary has no clean model.

5 Deep IP Protection: Invasive Solutions

5.1 Overview

DNN Watermarking is an invasive way for Deep IP protection. It embeds registrant-specified IP signatures $\boldsymbol{b}$ named watermarks into the target model $\mathcal{M}$ through model finetuning on a specific optimization objective about the secret key $\mathcal{K}$ and the watermarks $\boldsymbol{b}$. Given a correct key $\mathcal{K}$, the piracy models that finetune or extract from the target model $\mathcal{M}$ contain similar watermarks $\boldsymbol{b}'$: $dist(\boldsymbol{b}', \boldsymbol{b}) \rightarrow 0$. According to watermarks' host signals, the invasive solutions can be divided into the following types:

i) Inner-component-embedded Watermarking: Watermarks can be embedded into internal components of DNN models as hidden information, like model parameters, specific sparse structures, or neuron activation. It is usually realized by a regularization item *wrt* secret keys and watermarks of the loss function.

ii) Trigger-injected Watermarking: This way embeds watermarks through finetuning the target model on modified training sets with some trigger samples $\mathbf{X}_w$ as secret keys. The labels of $\mathbf{X}_w$, as the watermarks, are assigned by registrants and significantly different from the predicted labels of unwatermarked models.

iii) Output-embedded Watermarking: For generative models, watermarks can be embedded/extracted into/from normal outputs generated by target models without specific triggers. Different from traditional data watermarking, this watermarking process is hidden in target models. Here, watermark extraction relies on predefined rules or trained extractor DNNs. This way is usually used in image processing tasks or text generation tasks.

1,以norm的方式嵌入模型参数中；2，嵌入特定样本的输出中；3，嵌入所有输出中，如图片和文字中

Moreover, some schemes use a combination of these pathways or are designed for specific applied scenarios like self-supervised learning or transfer learning. We will describe them in this survey.

This section summarizes the representative methods for DNN watermarking in Table 3 from the following attributes:

- *Host Signals*: The carriers that watermarks embedded/extracted into/from, which we have discussed above.
- *Secret Keys*: Necessary inputs to extract watermarks, only with which the watermark can be extracted correctly.
- *Watermark Messages*: IP signatures embedded into the host signals of target models, like bit strings, labels, or data patterns.
- *Embedding Paradigms*: The main process to embed watermarks, including loss functions, training algorithms, or novel tricks.
- *Embedding Scenarios*: Watermarks can be directly embedded into the target model or indirectly embedded by watermarked model gradients in distributed training or federated learning.
- *Verification Scenarios*: The required model access for watermark extraction, including white-box, black-box and none-box. See Details in Section 3.2.
- *Target Networks*: The structure of target models, like fully-connected networks (FC), convolutional neural networks (CNN), Recurrent neural networks (RNN), etc.
- *Target Tasks*: The types of DNN models for different tasks or learning paradigms, such as classification or generated models.
- *Target Functions*: The IP functions aforementioned in Section 2.2 or their varieties, like copyright or integrity verification.
- *Datasets*: The type of tested datasets like images or texts.
- *Highlights*: We will emphasize the innovative highlights of research works from multiple perspectives, like motivation, technical improvements, target scenarios, etc.

修改模型嵌入水印

5.2 Inner-component-embedded Watermarking

Watermarks can be embedded into the following components:

5.2.1 Static Model Weights

Embedding watermarks into raw model weights. Uchida *et al* [19], [142] propose the first scheme for Deep IP Protection. Given the secret key matrix $\mathcal{K}$ and a binary signature $\boldsymbol{b}$, it appends an embedding regularizer $\mathcal{L}_R$ to the raw loss function $\mathcal{L}_0$ as Equ. 4, and embeds $\boldsymbol{b}$ by model finetuning with the loss $\mathcal{L}$,

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_0(\mathcal{X}, \boldsymbol{y}; \boldsymbol{\theta}) + \lambda \mathcal{L}_R(\mathcal{K}, \boldsymbol{b}; \boldsymbol{\theta}_w), \tag{4}$$

where $\boldsymbol{\theta}_w \subset \boldsymbol{\theta}$ is a parameter matrix selected from the whole model parameters $\boldsymbol{\theta}$. Here, $\mathcal{L}_R$ is a binary cross entropy function to make the extracted watermark $sign(\mathcal{K}\boldsymbol{\theta}_w)$ close to $\boldsymbol{b}$:

$$\mathcal{L}_R(\mathcal{K}, \boldsymbol{b}; \boldsymbol{\theta}_w) = \boldsymbol{b}\log(\sigma(\boldsymbol{\theta}_w\mathcal{K})) + (1-\boldsymbol{b})\log(1-\sigma(\boldsymbol{\theta}_w\mathcal{K})), \tag{5}$$

where $\sigma(\cdot)$ is a sigmoid function and $sign(\cdot)$ is a step function.

On this basis, many works are proposed for various demands. Chen *et al* design DeepMarks [20] for user managements, *i.e.*, embedding a unique watermark for each user's model and detecting undesired usages of distributed models. Its main threat is the Collusion Attack, that multiple attackers use their models to produce an unmarked model. To support massive users and detect collusion attackers, DeepMarks selects user signatures from an orthogonal matrix according to the coded modulation theory. Here, $\mathcal{L}_R$ is the mean square error function $\text{MSE}(\boldsymbol{\theta}_w\mathcal{K} - \boldsymbol{b}_j)$ where $\boldsymbol{b}_j$ is the signature of the j-th user.

Embedding watermarks into transformed model weights. Watermarks embedded into raw model weights are vulnerable to IP detection attacks and watermark overwriting attacks. To enhance the covertness of watermarks, a straightforward solution is to embed watermarks into some transformation of model weights. It is based on an assumption: if the weight

TABLE 3
The Comparison of Representative Methods for DNN Watermarking

	Methods	Year	Secret Key	Watermark Messages	Host Signals	Embedding Paradigms & Scenarios	Verification	Target Functions	Target Networks/Tasks	Data Scale & Type	Highlights
Inner-Component-Embedded Watermarks (WM)	Uchida *et al* [19]	2017	Secret Matrix	a unified bit string	raw model weights	finetune w/ CE regularizer loss; Directly	White-box	Copyright Verification	classification	small image datasets	The first scheme for Deep IP Protection
	Deepmarks [20]	2018	Secret Matrix	user-aware bit strings	raw model weights	finetune w/ MSE regularizer loss; -	White-box	User Management	classification	small image datasets	User Identification w/ Anti-collusion Coding
	Liu *et al* [96]	2021	RSA Public Key	a unified bit string	weights' greedy residuals	finetune w/ sign regularizer loss; -	White-box	Copyright Verification	CNN & RNN;	small; images&texts	robust WM w/o explicit ownership indicators
	RIGA [63]	2019	Secret Network	a unified bit string	DNN-transformed weights	co-train target, embedder and hider DNNs; -	White-box	Copyright Verification	classification	small/med image/text	Covert watermarks via a WM hider network
	Frequency WM [97] [98]	2020	Random seed for weights	a unified bit string	spectrum of weights	binarize spectrum & finetune non-watermark weights; -	White-box	Copyright Verification	classification	small image datasets	embed covert WM into the spectrum of weights
	Guan *et al* [99]	2019	WM Indices	a unified bit string	Model weights	embed WM by Histogram Shift; -	White-box	Integrity Verification	-	-	embed reversible watermarks to verify integrity
	NeuNAC [100]	2021	Secret Vector Set	a unified bit string	KLT basics of weights	embed WM into LSBs of weights; -	White-box	Integrity Verification	-	-	embed fragile watermarks to verify integrity
	DeepAttest [101]	2019	Secret Matrix	user-aware bit strings	Model weights	finetune w/ MSE regularizer loss;-	White-box	On-device Functionality	classification	small image datasets	Software-Hardware Co-design in TEE
	Deepsigns-W [21]	2018	Normal Samples & Secret Matrix	a unified bit string	hidden-lay activation	finetune w/ CE & MSE regularizer in activation isolation constraints; -	White-box	Copyright Verification	classification	small image datasets	embed WM into the exception of hidden-layer activations of some target classes
	MOVE-Gradient [102]	2022	Style-transferred Samples	meta-classifer	Model gradients	train the model on styled samples w/ CE loss;-	White-box	Copyright Verification	classification	small&medium; images	external features as WM verified by a meta-classifer
	Radioactive Data [56]	2020	Perturbed Samples	a shift vector	feature extractor	perturb dataset to create an isotropic shift on target feature extractor; -	White&Black	Dataset Protection & Copyright Verification	classification	medium; ImageNet	WM can be embedded from data to feature space
	Struct4WM [103] & [104]	2021	Secret Vector	a unified bit string	Model structures	channel/weight-wise model pruning; -	White-box	Copyright Verification	classification	small image dataset	Embed WM into model structures
	Chen *et al* [105]	2021	Key Mask Matrix	a QR code	Model structures	embed a QR code to the squeezed mask; -	White-box	Copyright Verification & Access Control	classification	small image datasets	the first to protect winning lottery tickets
	Watermark4NAS [106]	2021	fixed neural connections & hyperparameters		Model structures	search model structures constrained by watermarked neural connections; -	White&Black	Copyright Verification	CNN & RNN	small; images&texts	leverage watermarks to protect high-performance structures from NAS
	Passport-V1&2 [22], [23]	2019	Passport Matrix	a unified bit string	Scale & Shift of normalization layer	multi-task learning across passport-aware and passport-free model w/ sign regularizer loss & task loss; -	White-box	Copyright Verification & Access Control	classification	small&medium; images	Add a passport layer after each conv layer
	Passport-Norm [107]	2020	Passport Matrix	a unified bit string			White-box		classification	images & 3D points	Add a passport branch for verification
	RNN-IPR [108]	2022	Passport Matrix	a unified bit string	RNN Hidden State		White-box		RNN; Image Caption	small&medium; images	extend watermarks into RNNs
	HufuNet [109]	2021	Decoder DNN & HMAC key	an encoder DNN	Model weights	Iteratively training the encoder & target DNN w/ a robustness-assured loss; -	White-box	Copyright Verification	CNN & RNN; classification	small; texts&images	embed a subnetwork as watermarks verified by sample reconstruction
Trigger-Injected Watermarks (WM)	Adi *et al* [25]	2018	Unrelated Samples	Random Hard Labels	Model outputs	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	leverage backdoor attacks for Deep IP protection
	Deepsigns-B [21]	2018	Near OOD Samples	Target Hard Labels	Model outputs	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	select trigger samples from rarely explored regions
	Zhang *et al* [26]	2018	Embedded, Noised, or Unrelated Samples	Target Hard Labels	Model outputs	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	design various triggers for a proper robustness
	Guo *et al* [110]	2018	Samples w/ Message	Target Hard Labels	Model outputs	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	Inject an additive pattern with user message
	Maung *et al* [111]	2018	Block-wise Transform	Target Hard Labels	Model outputs	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	with no need of predefined trigger samples
	Zhu *et al* [112]	2018	Trigger Sample Chain	Target Hard Labels	Model outputs	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	use hashing to get a trigger chain against IP forge
	Few Weights [113]	2022	Samples w highly uncertain outputs	Target Hard Labels	Model outputs	finetune a few WM-sensitive and task-unrelated weights; -	Black-box	Copyright Verification	CNN& Transformer classification	small&medium; images	realize a proper tradeoff between model fidelity & WM robustness
	Frontier-Stitching [24]	2017	Adversarial Samples	Correct Hard Labels	Model outputs	get adversarial samples & finetune models; -	Black-box	Copyright Verification	classification	small image datasets	Repair adversarial samples for harmless WM
	EWE [114]	2021	Samples w/ SNNL-guided Optimized Trigger Patterns	Target Hard Labels	Model outputs	finetune w/ CE & SNNL loss alternately; -	Black-box	Copyright Verification	classification	small; image&speech	Entangle tasks and WM in the same channels for a high robustness against model extraction
	Blind WM [115]	2019	Samples w/ Hidden Logo	Target Hard Labels	Model outputs	finetune w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	Hide copyright logo into triggers by a GAN model
	DeepAuth [116]	2022	Near-boundary Samples	Correct Hard Labels	Model outputs	repair near-boundary adversarial samples; -	Black-box	Integrity Verification	classification	small image datasets	generate near-boundary samples as fragile WM
	UBW-P [117]	2018	Samples w/ a Pattern	Random Hard Labels	Model outputs	finetune or retrain w/ CE loss or inject poisoned samples into the dataset; -	Black-box	Copyright Verification Dataset Protection	classification	small&medium; images	Inject harmless trigger samples into the target model or the protected dataset
	New Label [118]	2020	OOD Trigger Samples	A New Hard Labels	Model outputs	retrain or finetune w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	treat crafted key samples as a new class
	Blackmarks [27]	2019	Signature-guided Trigger Samples	A bit string hidden in rearranged hard labels	Model Outputs	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	use class clustering for multi-bit trigger-based WM
	AIME [119]	2022	Samples predicted in error		Outputs' confusion matrix	finetune or retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small image datasets	class mispredictions as WM w/ a confusion matrix
	MOVE-Output [120]	2022	Style-transferred Samples	Clean-to-styled distances of Soft Predictions		retrain w/ CE loss; -	Black-box	Copyright Verification	classification	small&medium; images	embed external features into model outputs as WM
	CosWM [95]	2022	Subset of Training Samples	Specific Cosine Signals hidden in soft predictions		retrain w/ CE loss (modified soft labels);-	Black-box	Copyright Verification	classification	small image dataset	embed cosine signals with a predefined frequency to defend against ensemble distillation
	Namba *et al* [66]	2019	Trigger Samples	Target Hard Labels	Model Output	finetune exponentially-weighted weights; -	Black-box	Copyright Verification	classification	small image dataset	exponential weighting against IP detection & evasion
	Bi-level Opt [121]	2021	Samples w highly uncertain outputs	Target Hard Labels	Model Output	Optimize trivial weights like [113] and exemplars-to-be-embedded alternately;-	Black-box	Copyright Verification	classification	small&medium; images	Enhance model fidelity and watermark robustness by bi-level optimization on examplers and weights
	Target-Specific NTL [52]	2021	Samples w/ a specific patch	Prediction Accuracy on the target domain	Model Output	retrain w/ Inverted-IB loss to enlarge the source-target representation distances; -	Black-box	Copyright Verification	classification	small image datasets	introduce the inverted information bottleneck loss to enhance the watermark robustness
	Certified WM [122]	2022	Trigger Samples	Target Hard Labels	Model Output	alternately train raw & gaussian-noised weights	Black-box	Copyright Verification	classification	small&medium; images	Random smoothing on weights for certified WM
	UBW-C [117]	2022	Optimized Samples	Clean Hard Labels	Model Output	bi-level optimization on model and samples	Black-box	Dataset Protection	classification	small&medium; images	generate clean-label poison samples as watermarks
	Source-only NTL [52]	2021	Authorization Patterns	Prediction accuracy on non-source domains generated by a GAN framework		retrain w/ MMD-InvIB loss assisted by GAN-guided source domain augmentation	-	Applicability Authorization	classification	small image datasets	the first to propose the Deep IP function of applicability authorization
	M-LOCK [123]	2022		Low accuracy on samples w/o correct patterns		true labels only on samples w/ correct patterns	-		Classification	small&medium; images	train a pattern-sensitive decision boundary
Output	Zhang *et al* [124], [125]	2020	Extractor Subnetwork	IP patterns Invisible in output images	Output Images	Train a watermarked generator or an embedder after the generator jointly w/ a WM extractor	None-box	Copyright Verification & Tracing model abuse	image generation & processing models	small&medium; images	leverage watermarked images to watermark models
	Wu *et al* [126]	2020									
	Yu *et al* [127]	2021	Extractor Subnetwork	A bit string	Deepfake Outputs	train an encoder-decoder to watermark images		Deepfake Detection	Deepfake	small&medium; images	Watermarks are transferred from inputs to outputs
	AWT [128]	2022	Extractor Subnetwork	A bit string	Arbitrary Texts	jointly train hider/revealer/detector transformers	-	Text Watermarks	Text Generation	medium; text	a transformer for imperceptible and robust text WM
	WM4NLG [129] & [130]	2022	Predefined Rules	A bit string	Generated Texts	design some specific lexical rules of text generation	None-box	Copyright Verification	Text Generation	medium; text	watermark natural language generation models
Specific Scenarios	SDRW [131]	2022	Near-OOD Samples by cross-domain augmentation	Special Hard Labels	Model Outputs	finetune a selected core subnetwork for watermarking & embed extra reversible compensation information	White&Black	Copyright Verification & Integrity Verification	Transfer Learning	small image datasets	generalization and redundancy of core subnetworks for reversible and robust WM against transfer learning
	WAFFLE [132]	2020	Trigger Samples	Target Hard Labels	Model Outputs	finetune models after server-side aggregation; Directly transfer WM from local client models to the global server model via updated gradients; Indirectly global WM as IP signs & local WM for traceability; I local WM enhancing & global entangled aggregation; I	Black-box	Copyright Verification	Federated Learning	small image datasets	embed WM on the server side of a federated system
	FedIPR [133]	2021	Secret Matrix & Trigger Samples	Client-aware bit strings & trigger samples	Model outputs & norm-layer parameters		White&Black	Copyright Verification & Proof-of-work & Traceability		small image & text sets	embed WM on client sides w/ a theoretical analysis between WM capacities and detection rates
	FedTracker [134]	2022								small image datasets	introduce continual Learning to improve fidelity
	CITS-MEW [135]	2022	Trigger Samples	Target Hard Labels	Model Outputs		Black-box	Copyright Verification		small image datasets	multi-party entangled WM kept in global models
	GAN-IPR [136]	2021	Passport Matrix & Trigger Latent Codes	A unified bit string & an IP pattern in outputs	Norm-layer parameters & Generated Images	multi-task learning w/ SSIM & sign regularizer; Directly	White&Black	Copyright Verification	GAN models	small&medium; images	combine passports and backdoors to watermark models against IP ambiguity
	SSLGuard [137]	2022	Samples w/ a pattern & a verification decoder	IP vectors output from the verification decoder	Latent Space of Encoders	train watermarked/shadow encoders & a verification decoder jointly to embed detectable WM to latent codes	Black-box	Copyright Verification	Self-supervised Encoders	small image datasets	the first work to protect self-supervised encoders
	Sofiane *et al* [138]	2021	Trigger samples in 3 types	Modified Labels	Model Output	finetune models w/ task loss on trigger samples;	Black-box	Copyright Verification	extend WM to machine translation, regression, binary image classification, & reinforcement learning		
	CoProtector [139]	2022	Poisoned codes & comments w/ predefined rules		Output Codes/Comments	untargeted and targeted poisoning for illegal model detection and performance degradation	None-box	Code Repository Protection	Neural Code Tasks	medium code datasets	the first work to protect open-source code repositories
	SpecMark [140]	2020	Secret Matrix	A bit string	Spectrum of Weights	WM-guided binarization on spectrum of weights;-	White-box	Copyright Verification	Speech Recognition	small&medium; audio	the first work to watermark speech recognition models
Conflicting Interaction [141]		2022	analyze the conflicts of different strategies for model security via constrained multi-objective optimization on task/protection performance			model/dataset IP protection, adversarial training, DP-SGD				small image datasets	a framework to analyze conflicts & potential solutions

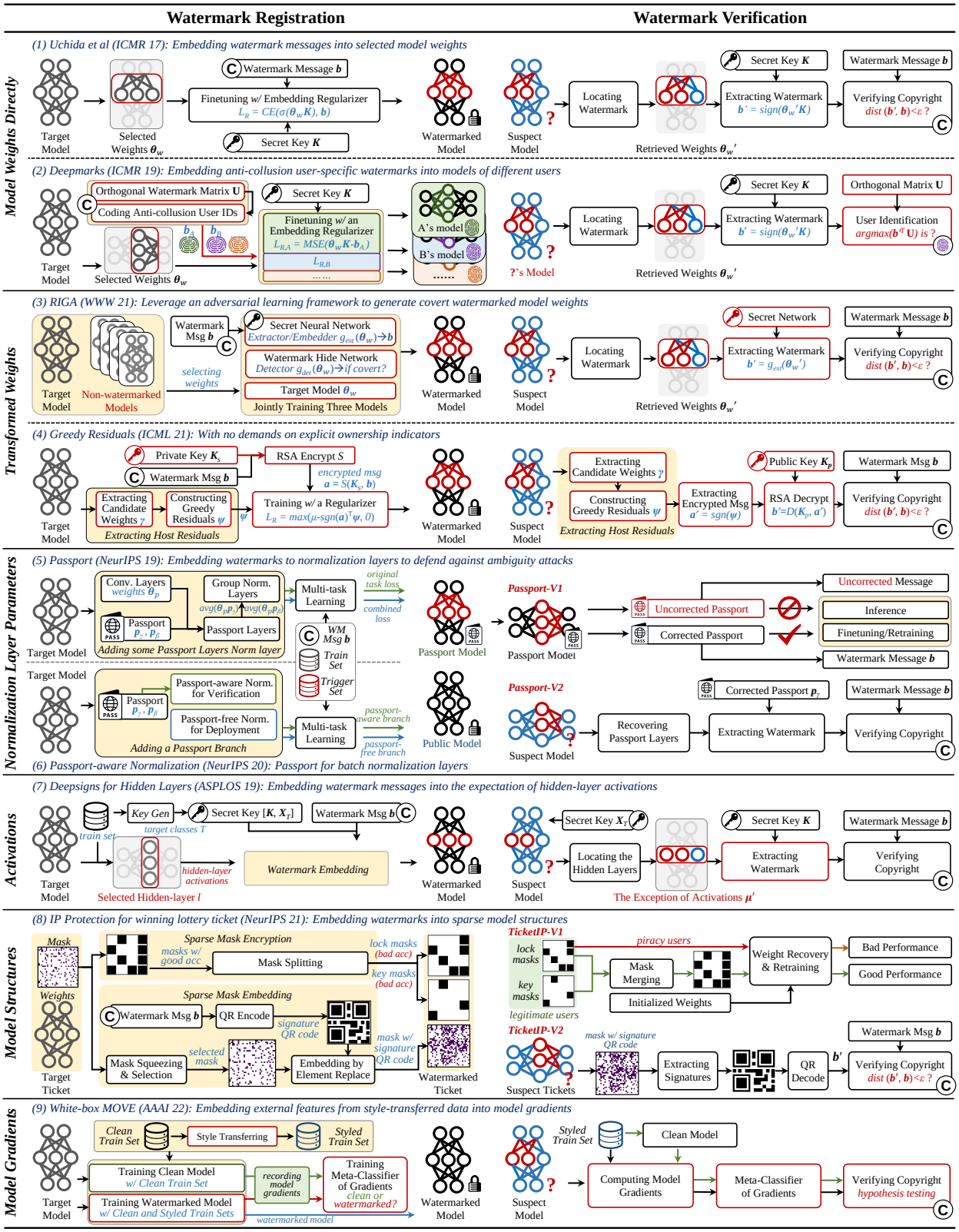


Fig. 8. The representative methods for inner-component-embedded watermarking. Watermarks can be embedded into model parameters, hidden-layer activations, model gradients and model structures.

修改不重要的模型参数，检测模型是否被篡改

distribution does not change significantly before and after watermark embedding, the watermarks can be considered covert.

i) Embedding watermarks into the frequency domain of weights can achieve this purpose to some extent: [97] and [98] randomly select the host model weights from multiple layers and compute the coefficients of the frequency domain components of the selected weights. Then, watermarks are embedded by quantifying the coefficients with the guide of the predefined IP signatures. The inverse transformation is used to obtain the watermarked weights to replace the selected weights. This method can avoid large amplitude changes in model parameters and reduce the impact of watermark embedding on weight distribution.

ii) DNNs can be also utilized to embed watermarks with more covertness: Considering robustness, covertness, and fidelity jointly, RIGA [63] designs an adversarial learning framework, including i) a generator network that outputs valid IP signatures given watermarked model weights otherwise outputs random signatures, used to extract or embed watermarks; ii) a detector network that tries to distinguish between watermarked and unwatermarked weights, used for watermark hiding.

iii) Specially-designed transformation rules can replace the role of predefined key matrices. IP ambiguity attacks, like forging attacks and overwriting attacks, can cause copyright conflict by forging secret matrices (see details in Section 4.2). Against IP ambiguity attacks, Liu *et al* follow the "less is more" principle, and design Greedy Residuals [96] with no demands on explicit ownership indicators like trigger samples or secret matrices. Here, watermarks are embedded into a residual vector ψ (extracted from important weights) by a regularizer $\mathcal{L}_R$ defined as:

RSA加密部分权重，记录秘文的符号

$$\mathcal{L}_R = \max(\alpha - \text{sgn}(\boldsymbol{b}_{enc})^{\mathsf{T}}\boldsymbol{\psi}, 0)$$

where the n-bit IP signature $\boldsymbol{b}_{enc}$ is a bit string, *i.e.,* an encrypted message by the RSA algorithm; sgn:$\{(0,+\infty],[-\infty,0]\}\rightarrow\{1,-1\}$ is a function to get the signs of input values. The regularizer $\mathcal{L}_R$ encourages the elements of $\boldsymbol{\psi}$ to stay the same sign as that of $\text{sgn}(\boldsymbol{b}_{enc})$. The preset threshold α is to enlarge the gap between different signs of values. The residual $\boldsymbol{\psi}$ is equivalent to the mean of important weights: First, a $1d$-average pooling is applied to transform reshaped weights $\theta^l \in \mathbb{R}^{n\times d\times m}$ of the l-layer to $\gamma^l \in \mathbb{R}^{n\times d}$. Then for each row in γ^l, select some elements with the largest absolute values and compute their mean values, and finally compose the residual vector $\boldsymbol{\psi} \in \mathbb{R}^n$.

Weight-based watermarks for integrity verification. Watermarks can be designed to be reversible or fragile to realize integrity verification against model tampering attacks [99]–[101], [143].

可逆转的

i) Reversible watermarks. Since irreversible watermarking modifies the network contents permanently and destroys the model integrity, Guan *et al* develop a reversible watermarking algorithm [99] for integrity verification of CNN models inspired by channel pruning and histogram shift. The "reversible" means that the raw target model can be recovered by erasing the watermark of the watermark model. In this way, it only needs to compare the hash codes (e.g., SHA-256) of the recovered model and the raw target model to determine whether the model has been modified. [99] is mainly suitable for tampering detection during model distribution. First, it selects the weights $\boldsymbol{\theta}_w$ of k unimportant network channels with minimum channel entropy values as the host sequence. Given the l-th layer with d-channels, the process to compute the recovered channel entropy $\{H_i|i=1...d\}$ is: ① getting the l-th layer's activation tensor $\mathcal{A}^l \in \mathbb{R}^{|\mathbf{X}_w|\cdot d\cdot h\cdot w}$ on some samples $\mathbf{X}_w$; ② using global average pooling to transform $\mathcal{A}^l$ into a matrix $\hat{\mathcal{A}}^l \in \mathbb{R}^{|\mathbf{X}_w|\cdot d}$; ③ using the histogram estimation to establish the probability distribution $\boldsymbol{p}_i$ for each channel (each column of $\hat{\mathcal{A}}^l$) and computing the entropy H_i for each $\boldsymbol{p}_i$. Then, the float host sequence $\boldsymbol{\theta}_w$ will be transformed to the integer host sequence $\hat{\boldsymbol{\theta}}_w$ with the minimum entropy among all possible integer sequences by intercepting two digits (in the range [-99,99]) from each float element of $\boldsymbol{\theta}_w$. Finally, the integer host sequence $\hat{\boldsymbol{\theta}}_w$ can be considered as a grayscale image and watermarks can be embedded via image reversible data hiding strategy like the histogram shift strategy.

ii) LSB-based Fragile watermarks. NeuNAC [100] embeds the fragile watermark by modifying the least significant bytes (LSB) of model parameters. It can provide tampering detection and localization during the model execution. It divides model parameters into many Parameter Units (PUs) with 16 parameters. Each parameter is a 4-byte float number, where the first three bytes denote the most significant bytes (MSB) and the last byte denotes the LSB. A 32-byte Watermark Embedding Unit (WEU) is constructed for each PU, where each byte of the first 16 bytes is the hash code of the corresponding MSB and each byte of the last 16 bytes is the corresponding LSB. The watermark message $\boldsymbol{b}$ is embedded piecewise into the coefficients of Karhunen-Loève Transform (KLT) of each WEU while minimizing the change of LSBs by Genetic Algorithms, where the secret key $\mathcal{K}$ is the basic vectors of KLT. The verification process on WEUs can be performed in parallel, and the model is tampered with once any segment of the extracted watermark does not match the corresponding embedded watermark segment.

iii) Fragile watermarks for on-device functionality verification. The above methods do not allow any change on watermarked weights, however, their computation cost hinders the on-device verification. Some scenarios only require functionality verification because a slight model modification imposed by the attackers, such as bit-inversion attacks, can severely affect the model. Based on DeepMarks [20], Deepattest [101] embeds device-specific watermarks to the target models and provides an software-hardware co-design approach to deploy the verification process in a Trusted Execution Environment (TEE). Once the functionality verification fails, the currently running model will be blocked. The considered threats mainly include ① Full-DNN Program Substitutions that attackers map illegitimate models to the protected device; ② Forgery Attacks that attackers forge the device-specific watermarks in the secure memory to mislead the verification; ③ Fault Injection that attackers modify the model contents stored in the memory at a fine-grained level.

5.2.2 Dynamic Hidden-layer Activation or Gradients

Dynamic hidden-layer activations. Parameter-based watermarks are vulnerable to overwriting attacks that insert new watermarks in the same way to stain embedded watermarks. Thus, DeepSigns [21] watermarks the expectation $\boldsymbol{\mu}$ of hidden-layer activations on target classes T via a regularizer $\mathcal{L}_R$ *wrt* the secret matrix $\mathcal{K}$ and T-class data $\mathbf{X}_T$, as

$$\mathcal{L}_R = \mathcal{L}_{\text{CE}}(\sigma(\boldsymbol{\mu}\mathcal{K}), \boldsymbol{b}) + \beta\sum_{i\in T}\|f_i(\mathbf{X}_T) - \boldsymbol{\mu}_i\| - \beta\sum_{i\in T, j\notin T}\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|, \quad (7)$$

where the first item is to transform the output exception $\boldsymbol{\mu}$ of the target classes T into the watermark $\boldsymbol{b}$; the second item is to minimize the variance of the output activation $f_T(\mathbf{X}_T)$ on target classes T; and the third item is to maximize the divergence between watermarked classes and unwatermarked classes.

Dynamic Gradients. Experiments conducted in [120] show that, in the way of backdoor watermarking, the predictions on predefined trigger samples could not be matched between the watermarked model and its piracy version. Moreover, extra security risks like backdoors will be introduced. Fingerprinting

methods like dataset inference [144] are possible to make misjudgments and provide incredible results. Towards an effective and harmless watermarking method, MOVE [102], [120] embeds external features from style transfer into the target model, instead of changing the labels of trigger samples. For white-box verification, the external features are depicted by model gradients on trigger samples. It mainly consists of three steps: i) generating some trigger samples $\mathbf{X}_w$ by style transfer on the training set $\mathbf{X}$; ii) training a watermarked model $\mathcal{M}_w$ on $\{\mathbf{X}_w, \mathbf{X}\}$ and an unwatermarked model $\mathcal{M}$ on $\mathbf{X}$; iii) getting the gradients of $\mathcal{M}_w$ and $\mathcal{M}$ on $\mathbf{X}_w$, and training a binary meta-classifier to distinguish the gradients from $\mathcal{M}_w$ and $\mathcal{M}$. Since model gradients cannot be obtained in black-box scenarios, MOVE adopts the difference of soft predictions between augmented style-transferred samples and their raw samples, as the substitution of model gradients. Finally, hypothesis testing is performed on the predictions of the trained meta-classifier to obtain the verification result.

For dataset protection: watermarks from feature space to data space. Watermarks can be embedded into the feature space by adding additive patterns to training samples. This approach does not require a specific model training process and can therefore be used to protect not only the model ownership but also the dataset ownership. That is, to identify the models illegally trained on the protected dataset. Radioactive Data [56] is a typical method for such studies. It represents the model as the combination of a linear classifier $\boldsymbol{w}$ and a feature extractor ϕ, *i.e.*, $\boldsymbol{w}\phi(\boldsymbol{x})$. Its watermark embedding stage aims to embed a random isotropic unit vector $\boldsymbol{u}_i$ into the feature space of all training samples of each class i, *i.e.*, the output of the feature extractor $\phi(\boldsymbol{x})$. This is done through pixel-level optimization on training samples as

$$\min_{\hat{\boldsymbol{x}}} -(\phi(\boldsymbol{x}) - \phi(\hat{\boldsymbol{x}}))^\top \boldsymbol{u} + \lambda_1 \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 + \lambda_2 \|\phi(\boldsymbol{x}) - \phi(\hat{\boldsymbol{x}})\|_2, \quad (8)$$

where the first term is to align the features with $\boldsymbol{u}$ while the last two items are to minimize the impact of embedded patterns on training samples and their features. Given a suspect model $\boldsymbol{w}_s\phi_s(\boldsymbol{x}) \approx \boldsymbol{w}_s\boldsymbol{A}\phi(\boldsymbol{x})$, the cosine similarity $\cos(\boldsymbol{u}, \boldsymbol{w}_s\boldsymbol{A})$ should follow the beta-incomplete distribution if the model is trained on normal samples $\boldsymbol{x}$, otherwise, if the model is trained on protected data $\hat{\boldsymbol{x}}$ or is finetuned from the protected model $\boldsymbol{w}\phi(\boldsymbol{x})$, the cosine similarity will be significantly higher. This method can be extended to black-box verification by extract the surrogate model from the suspect model.

5.2.3 Model structures

Model structures for watermarks. Watermarked sparse model structures are exploited to protect Deep IP Protection. Some methods [103], [104] apply model pruning to embed watermarks into model structures rather than parameters to bypass IP removal attacks using parameter modification. [103] applies channel pruning for watermark embedding. The IP signature is piecewise embedded into the channel pruning rates of multiple convolutional layers and the watermarked model will be obtained by channel pruning using the given sequence of watermarked pruning rates. [104] considers weight-level pruning and embeds watermarks into the model weights with low connection sensitivity. However, these methods cannot defend against channel pruning and model extraction using different model structures obviously.

Watermarking model structures. The high-performance model structure itself is also worth protecting. The lottery ticket hypothesis [145] has demonstrated that the model trained on a subnetwork with a specific sparse mask and initial weights, *i.e.*, a winning ticket, can provide accuracy close to the full network while significantly reducing inference and training costs. Thus, winning tickets that are costly to find can be regarded as valuable assets, and Chen *et al* explore a new paradigm to protect the ownership of winning tickets [105]. The l-layer winning ticket is defined as $\mathbf{M}^l \odot \boldsymbol{\theta}_0^l$, where $\boldsymbol{\theta}_0^l$ and $\mathbf{M}^l$ denote the weights and the sparse mask of the l-layer respectively, and $\odot$ denotes the Hadamard product, *i.e.*, element-wise multiplication. [105] splits the mask with good accuracy into a key mask $\mathbf{M}^l_{\text{key}}$ and a lock mask $\mathbf{M}^l_{\text{lock}}$ with bad accuracy, where $\mathbf{M}^l = \mathbf{M}^l_{\text{key}} + \mathbf{M}^l_{\text{lock}}$. Only the legitimate user with both the key mask and the lock mask can train a good model. A QR code containing the IP signature is also embedded into the transformed sparse mask by element replacement to support white-box copyright verification. Neural Architecture Search (NAS), which automatically discovers a good network structure for a given task and dataset, provides neural network structures with higher accuracy than human-designed structures while consuming huge computation resources. Thus, Lou *et al* [106] treat the high-performance network structures derived by NAS as an important asset and propose to embed watermarks into the searched model structures. [106] creates the owner-specific watermarks by fixing some selected connections in the search space and then searches the rest space to derive a high-performance network structure. To support the black-box verification, the authors design a more comprehensive method based on cache side-channel analysis to extract model structures by a side-channel pattern.

5.2.4 Extra components

Watermarks can be embedded by injecting an extra process or a component into the target model. Unlike the watermarking schemes described before, this category of methods change the process of model inference via extra steps and thus some of them support access control. Next, we introduce some typical methods:

Extra processing on normalization layers. Watermarking methods independent of the original task loss are vulnerable to IP ambiguity attacks. To overcome this limitation, Passport [22], [23] appends a passport layer after each selected hidden layer; then the target model can output valid predictions or be finetuned only with the correct passport $\mathcal{P} = \{\mathbf{P}^l_\gamma, \mathbf{P}^l_\beta | l = 1...L\}$. The l-th passport layer is defined as:

$$\begin{aligned} f^l_{\text{pass}} &= \gamma^l(\mathbf{P}^l_\gamma) \cdot f^l(\mathbf{X}^l) + \beta^l(\mathbf{P}^l_\beta), \ f^l(\mathbf{X}^l) = \boldsymbol{\theta}^l_p \otimes \mathbf{X}^l \\ \gamma^l(\mathbf{P}^l_\gamma) &= avg(\boldsymbol{\theta}^l_p \otimes \mathbf{P}^l_\gamma), \ \beta^l(\mathbf{P}^l_\beta) = avg(\boldsymbol{\theta}^l_p \otimes \mathbf{P}^l_\beta) \end{aligned} \quad (9)$$

where $\gamma^l(\mathbf{P}^l)$ and $\beta^l(\mathbf{P}^l_\beta)$ are the scale and bias parameters of normalization layers and computed using the passport $\mathcal{P}$; $\boldsymbol{\theta}^l_p$ denotes the parameters of the l-th hidden layer; $\otimes$ denotes the operation of the hidden layer (e.g., convolution or multiplication). Furthermore, IP signatures could also be embedded into some scale and bias parameters via a sign loss as

$$\mathcal{L}_R = \max(\alpha - \boldsymbol{\gamma}^\mathsf{T} \text{sgn}(\boldsymbol{b}), 0), \quad (10)$$

where $\boldsymbol{\gamma}$ is the scale vector selected to embed the IP signature $\boldsymbol{b}$ in the same shape as $\boldsymbol{b}$. It encourages $\boldsymbol{\gamma}$ to have the same sign as $\text{sgn}(\boldsymbol{b})$. Model owners can distribute the passport $\mathcal{P}$ to legitimated users along with the target model, or joint train the passport branch and the normal branch via multi-task learning and distribute the normal branch. Trigger samples can also be embedded into the model for black-box verification. However, batch normalization cannot support such multi-task learning, and group normalization will degrade model accuracy. Therefore, zhang *et al* propose passport-aware normalization [107] that

supports most existing normalization layers and only requires an extra passport-aware branch, which is represented as

$$\hat{f}^l = \begin{cases} \gamma_0 \frac{f^l(\mathbf{X}^l)-\mu_0(f^l(\mathbf{X}^l))}{\sigma_0(f^l(\mathbf{X}^l))} + \beta_0, & \text{passport-free,} \\ \gamma_1(\mathbf{P}_\gamma^l) \frac{f^l(\mathbf{X}^l)-\mu_1(f^l(\mathbf{X}^l))}{\sigma_1(f^l(\mathbf{X}^l))} + \beta_1(\mathbf{P}_\beta^l), & \text{passport-aware,} \end{cases} \tag{11}$$

where $\gamma_1(\mathbf{P}_\gamma^l) = \boldsymbol{\theta}_2 \cdot h(\boldsymbol{\theta}_1 \cdot avg(\boldsymbol{\theta}_p^l \otimes \mathbf{P}_\gamma^l))$, $\beta(\mathbf{P}_\beta^l) = \boldsymbol{\theta}_2 \cdot h(\boldsymbol{\theta}_1 \cdot avg(\boldsymbol{\theta}_p^l \otimes \mathbf{P}_\beta^l))$ are output from two fully connected layers with the weights $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. Different from [22], [23], here the mean/std statistics are computed separately for the passport branch μ_1, σ_1 and the normal branch μ_0, σ_0, thereby supporting multi-task learning. Moreover, the learnable weights to transform passports reduce the accuracy degradation caused by the passport branch.

Extra processing on RNN hidden states. [108] extends watermarks to recurrent neural networks (RNN). With reference to [23], Lim *et al* realize access control and copyright verification for both black-box and white-box scenarios simultaneously. To be specific, the protected model is trained using the secret key $\mathcal{K}$ to transform the raw hidden state $\boldsymbol{h}$ into $\hat{\boldsymbol{h}}$ after each time step as

$$\hat{\boldsymbol{h}} = \mathcal{K} \otimes \boldsymbol{h}, \ \mathcal{K} = BE \odot BC \tag{12}$$

where BE is the binary representation of the owner-defined key string, $BC \in \{-1, 1\}^{|\mathcal{K}|}$ is a random sign vector, $\hat{\boldsymbol{h}}$ is the output hidden state at the time step. Here, $\odot$ denotes element-wise multiplication, and $\otimes$ can be selected as element-wise multiplication or element-wise addition. Similar to [23], an IP signature $\boldsymbol{b}$ can be embedded into the hidden state $\boldsymbol{h}$, using the sign loss regularization item as

$$\mathcal{L}_R = \max(\alpha - \boldsymbol{h}^\mathsf{T} \text{sgn}(\boldsymbol{b}), 0). \tag{13}$$

The loss forces the hidden state $\boldsymbol{h}$ to have the same sign as $\text{sgn}(\boldsymbol{b})$. Note that it has three verification ways same as [23].

Subnetworks as Watermarks. Most watermarking methods based on inner-components cannot theoretically guarantee robustness against IP removal attacks. To this end, lv *et al* [109] delicately design a loss function for watermark embedding to theoretically guarantee that the accuracy of the task model is more sensitive to fine-tuning attacks than that of the watermark subnetwork. Moreover, through such subnetwork-based watermark embedding, [109] achieves the watermark stealthiness and forgery resistance to a certain extent. It first trains the Hufunet, an encoder-decoder model for data reconstruction on trigger samples. All the encoder's parameters, as the watermarks, are embedded into the target model where the embedding location of each parameter is determined by the hashcode *wrt* its mirrored parameter in the decoder, the secret key, and the parameter index. Then it alternately trains the embedded watermark subnetwork and the target model to ensure the accuracy of both the subnetwork g_w and the target model g. The loss function is defined as:

$$\begin{aligned} \mathcal{L} = \mathcal{L}_{\text{task}} &+ \left(\frac{\mathbb{E}[\nabla_g \mathcal{L}_{\text{task}}]}{\mathbb{E}[\nabla_{g_w} \mathcal{L}_{\text{task}}]} - \alpha_1 \right)^2 + \left(\frac{\mathbb{E}[\nabla_g \text{PCC}]}{\mathbb{E}[\nabla_{g_w} \text{PCC}]} - \alpha_2 \right)^2 \\ &+ \left(\frac{var[\nabla_g \mathcal{L}_{\text{task}}]}{var[\nabla_{g_w} \mathcal{L}_{\text{task}}]} - \alpha_3 \right)^2, \\ \text{s.t.} \quad & \alpha_1 \cdot \alpha_2 > 1, \ \ \alpha_3 < \alpha_1^2, \end{aligned} \tag{14}$$

where PCC is the normalized loss change due to the attack; $\alpha_1, \alpha_2, \alpha_3$ are preset constants. It ensures that the target model fails before the watermark subnetwork under fine-tuning attacks. The decoder is saved by the verifier. During verification, the verifier extracts an encoder from the suspect model, and merges the extracted encoder and the decoder into a complete HufuNet. It is considered as an infringement if the reconstruction performance on trigger samples is higher than a threshold.

5.3 Trigger-injected Watermarking

Most watermarking methods based on trigger samples are designed for black-box verification and have three main points: i) generating trigger samples $\mathbf{X}_w$ as the secret keys of watermarks; ii) labeling trigger samples, *i.e.*, determining the expected output $\boldsymbol{y}_w$ of $\mathbf{X}_w$ on the watermarking model $\mathcal{M}_w$ significantly different from that on irrelevant models $\mathcal{M}_-(\mathbf{X}_w)$; and iii) embedding paradigms, *i.e.*, how to make the output $\mathcal{M}_w(\mathbf{X}_w)$ close to $\boldsymbol{y}_w$ while meeting the demands on fidelity and discriminability.

$$\mathcal{L}([\mathcal{X}, \mathbf{X}_w], [\boldsymbol{y}, \boldsymbol{y}_w]) = \mathcal{L}_0(\mathcal{X}, \boldsymbol{y}) + \lambda \mathcal{L}_w(\mathbf{X}_w, \boldsymbol{y}_w). \tag{15}$$

5.3.1 Generating trigger samples

Trigger samples can be selected from datasets directly or simply processed versions. Here, we introduce some typical cases:

- *Out-Of-Distribution (OOD) samples as trigger samples*: Adi *et al* [25] propose the first trigger-based watermarking method. Here, the trigger samples are abstract, out-of-distribution images with randomly assigned labels from uncorrected classes. Assigning target hard labels is also feasible [26]. Trigger samples can also be selected according to the OOD degree, *i.e.*, the position (or likelihood) in the data distribution [21], [131]. For example, Deepsigns [21] proposes to select trigger samples from rarely explored regions where each sample has few neighbors within an ε-ball of activations. SRDW [131] is to choose an appropriate OOD degree to trade off fidelity and robustness.
- *Samples with highly uncertain prediction results [113]*: Such samples, whose soft predictions are close across classes, are usually nearby the decision boundary of the target model. Their predictions are easy to be manipulated by minor model modifications. These samples can be embedded while ensuring fidelity. Generally, it can be measured by the Shannon Entropy of the model predictions.
- *Random noises as trigger samples*: Noises of the same dimension as input samples can also be used as trigger samples, like noises sampled from a Gaussian distribution [26], [138].
- *Natural samples attached with secret, specific additive trigger patterns*: Most trigger-based watermarking schemes [26], [110], [114] apply this type of trigger sample, which can generally be formalized as $\boldsymbol{x}_w = (1-\alpha)\boldsymbol{x} + \alpha \boldsymbol{t}$ or $\boldsymbol{x}_w = (1-\Gamma) \odot \boldsymbol{x} + \Gamma \odot \boldsymbol{t}$. Here, $\boldsymbol{t}$ denotes the trigger pattern, $\alpha \in (0, 1]$ is a hyperparameter, and Γ is a mask matrix to select the position of the trigger pattern. The trigger pattern can be specific content like a string covering part of images, or be random noises like Gaussian noise. Moreover, such additive patterns can be also designed to imply the copyright information of model owners [110].
- *A chain of trigger samples generated by a hash function*: An attacker can forge trigger samples and their labels by means of adversarial attacks. To disable forged trigger samples, Zhu *et al* [112] propose to execute the hash function multiple times on seed samples to generate a chain of trigger samples with assigned labels. In this way, attackers cannot construct a chain of trigger samples without model finetuning.
- *Block-wise processing*: Trigger samples can be generated on arbitrary training samples through predefined operations without being specified in advance. Maung *et al* [111] design a way of block-wise transformation associated with a predefined secret key to achieve this goal.

Copyright information of owners can be embedded into trigger samples via generative models. Zheng *et al* leverage a generative-adversarial framework to generate blind trigger samples embedded with the owner's logo [115]. Here, the "*blind*" refers that the generated trigger samples can defend against IP

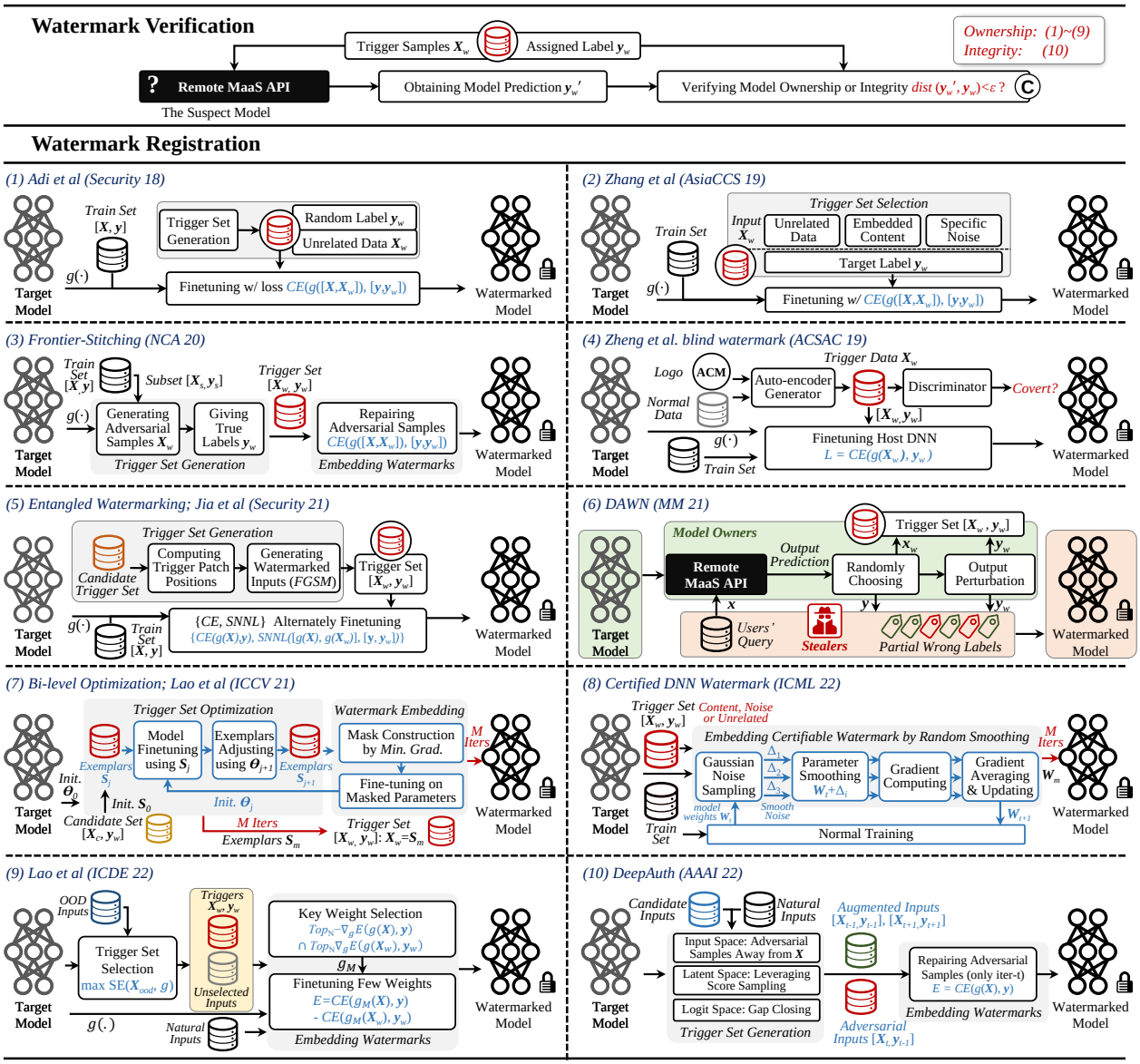


Fig. 9. The representative methods for trigger-injected watermarking.

detection attacks. This framework consists of two DNN models: i) a trigger generator that hides the owner-specific logo into natural samples and outputs the generated trigger samples $\mathbf{X}_w$, and ii) a discriminator that tries to detect the embedded content from $\mathbf{X}_w$.

Synthetic samples can also be used as trigger samples, like adversarial samples or generated samples. Adversarial samples created by adding carefully-crafted small perturbations Δ to seed samples $\mathbf{X}_0$ induce the target model to produce incorrect predictions. Given some seed samples with the labels $\boldsymbol{y}_0$, the adversarial samples $\mathbf{X}_w$ could be generated by untargeted adversarial attacks that make $g(\mathbf{X}_w) \neq \boldsymbol{y}_0$ as

$$\max_{\Delta} \mathcal{L}_{\text{CE}}(g(\mathbf{X}_w), \boldsymbol{y}_0), \ \mathbf{X}_w = \mathbf{X}_0 + \Delta, \ \|\Delta\| \leq \epsilon, \tag{16}$$

or targeted attacks to produce the predefined results $\boldsymbol{y}_w$, as

$$\min_{\Delta} \mathcal{L}_{\text{CE}}(g(\mathbf{X}_w), \boldsymbol{y}_w), \ \mathbf{X}_w = \mathbf{X}_0 + \Delta, \ \|\Delta\| \leq \epsilon, \ \boldsymbol{y}_w \neq \boldsymbol{y}_0, \tag{17}$$

where ϵ is to limit the magnitude of Δ. Equ. 16 can be solved by the Fast Gradient Sign Method (FGSM) as

$$\boldsymbol{x}_w = \boldsymbol{x} + \epsilon \cdot \text{sgn}(\nabla_x \mathcal{L}_{\text{CE}}(g(\boldsymbol{x}), y_0)), \forall \boldsymbol{x} \in \mathbf{X}_0 \tag{18}$$

where $\boldsymbol{x}_w$ is the adversarial sample of $\boldsymbol{x}$, or be solved by Projected Gradient Descent (PGD), an iterative method whose t-step is

$$\boldsymbol{x}^{t+1} = \prod_{B_\epsilon} (\boldsymbol{x}^t + \alpha \cdot \text{sgn}(\nabla_x \mathcal{L}_{\text{CE}}(g(\boldsymbol{x}), y_0))), \forall \boldsymbol{x}^0 \in \mathbf{X}_0 \tag{19}$$

where sgn:$\{(0, +\infty], [-\infty, 0]\} \to \{1, -1\}$ is the function to get the signs of input values. $\prod_{B_\epsilon}(\cdot)$ is an operation to project the input to a small range B_ϵ, a ball with the center $\boldsymbol{x}^0$ and the radius ϵ, to limit the perturbation. The step length α is much smaller than ϵ.

Frontier-Stitching [24] is the first work to introduce adversarial samples into trigger-injected watermarking. It generates adversarial samples by untargeted FGSM as trigger samples $\mathbf{X}_w$, where some samples successfully change their predictions but others do not. Then, it select the true labels $\boldsymbol{y}_0$ of $\mathbf{X}_w$ as the IP signature and fine-tunes the target model on $\{\mathbf{X}_w, \boldsymbol{y}_0\}$. We name this process repairing adversarial samples.

Aiming at the problem that trigger samples are easy to be removed by attackers, Jia *et al* analyzed: since task samples and trigger samples are learned by different neurons, finetuning the watermarked model on task samples can greatly change the neurons that learn the trigger samples. Thus, Jia *et al* propose Entangled Watermarking Embeddings (EWE) [114] so that both the task and trigger samples are learned by the same neuron group, based on the soft nearest neighbor loss (SNNL) defined as

$$\mathcal{L}_{\text{SNNL}}(\mathcal{X}, \mathcal{Y}, T) = -\underset{\boldsymbol{x}_i \in \mathcal{X}}{\text{mean}}\{\log(\frac{\sum_{j \neq i, y_i = y_j} e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/T}}{\sum_{k \neq i} e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2/T}})\}, \tag{20}$$

where T denotes the temperature generally decreasing with the

number of iterations. The SNNL depicts the entanglement of data manifolds, and maximizing $\mathcal{L}_{\text{SNNL}}$ is equivalent to making the representation distances between samples of different classes close to the average representation distance of all samples. Based on the SNNL, EWE first selects the position of trigger patches on trigger samples with the maximal gradients, and then adds the adversarial perturbation onto trigger samples using the untargeted FGSM on the cross-entropy loss $\mathcal{L}_{\text{CE}}$ and the SNNL $\mathcal{L}_{\text{SNNL}}$. Finally, EWE embeds the trigger samples $\mathbf{X}_w$ with the labels $\boldsymbol{y}_w$ using the loss as Equ. 21. Here, EWE computes the SNNL on each layer g_l of the target model $\mathcal{M}$.

$$\mathcal{L}=\mathcal{L}_{\text{CE}}(\mathbf{X},\boldsymbol{y})-\alpha\sum_{\forall l\in\mathcal{M}}\mathcal{L}_{\text{SNNL}}([g_l(\mathbf{X}_0),g_l(\mathbf{X}_w)],[\boldsymbol{y}_0,\boldsymbol{y}_w],T_l). \quad (21)$$

Trigger-based fragile watermarks. Trigger-based watermarks can be designed to be fragile for integrity verification [116], [146], [147]. The core idea is to generate trigger samples whose predictions are sensitive to any model modification. Deepauth [116] is a typical scheme of trigger-based fragile watermarks. It adopts a similar idea to Frontier-Stitching [24], *i.e.*, repairing adversarial samples, while the difference is that Deepauth generates fragile trigger samples closer to the decision boundary but far away from natural samples, and thus the decision boundary of the watermarked model will shift only a little bit to modify the predictions of the fragile trigger samples just enough. The "fragile" refers that a slight model modification will change the predictions of trigger samples. The process for generating trigger samples mainly consists of three steps: i) from input space, Deepauth seeks the adversarial samples $\mathbf{X}_{w,1}$ that can change the predictions only when the l_p distance $l_p(\mathbf{X},\mathbf{X}_{w,1})$ is larger than a threshold ϵ_d, as

$$\begin{aligned}&\min_{\mathbf{X}_{w,1}}\ \|\mathcal{L}_{\text{CE}}(g(\mathbf{X}_{w,1}),\boldsymbol{y}_w)-\mathcal{L}_{\text{CE}}(g(\mathbf{X}_w),\boldsymbol{y})\|,\\ &\text{s.t.}\quad \boldsymbol{y}_w=g(\mathbf{X}_{w,1})\neq\boldsymbol{y},\ l_p(\mathbf{X},\mathbf{X}_{w,1})\geq\epsilon_d;\end{aligned} \quad (22)$$

ii) from latent space, Deepauth applies leverage score sampling to estimate the uncertainty of $\mathbf{X}_{w,1}$ and selects the samples with the largest leverage scores as the trigger samples $\mathbf{X}_{w,2}$ towards a minimal impact on fidelity; and iii) from logit space, the samples $\mathbf{X}_{w,2}$ are further moved to the decision boundary by reducing the logit difference between the classes $\boldsymbol{y}_w$ and $\boldsymbol{y}$, via an iterative gradient optimization where the t-step is expressed as

$$\mathbf{X}_{t+1}\leftarrow\mathbf{X}_t+\alpha\cdot\text{sgn}(\nabla_{\mathbf{X}_t}(\mathcal{L}(g(\mathbf{X}_t),\boldsymbol{y}_w)-\mathcal{L}(g(\mathbf{X}_t),\boldsymbol{y}))), \quad (23)$$

where α is the step length and $\mathbf{X}_0$=$\mathbf{X}_{w,2}$. The iteration step just changing predictions is recorded as t^*; then $\mathbf{X}_{t^*}$ is the trigger samples, given which the raw target model outputs $g(\mathbf{X}_{t^*})=\boldsymbol{y}_w$. Then the model is fine-tuned on the trigger set $[\mathbf{X}_{t^*},\boldsymbol{y}]$ along with the augmented set $[\mathbf{X}_{t^*-1},\boldsymbol{y}]\cup[\mathbf{X}_{t^*+1},\boldsymbol{y}_w]$ to embed trigger samples while bounding the decision boundary shift.

Trigger samples for dataset ownership protection. Dataset ownership protection raises new requirements for trigger sample generation: First, the injection of trigger samples should not introduce additional security risks for models trained on protected datasets. Second, the injected trigger samples should have stealthiness and cannot be detected by the dataset users. Finally, the models trained on the protected dataset should contain the IP signature of the dataset owner. However, targeted backdoor watermarking injects poison backdoors, *i.e.*, trigger samples with specific target labels, into the target model. Attackers can add the trigger pattern to an arbitrary input sample to deterministically manipulate the model prediction.

Thus, Li *et al* propose Untargeted Backdoor Watermarks (UBW) [117], a harmless and stealthy watermarking method where watermarked models output randomly predicted labels with low confidences on trigger samples instead of specified target labels. UBW considers the three demands for dataset protection and it can be naturally extended to model ownership protection. A natural thought to maximize the dispersibility is randomly shuffling the true labels of trigger samples (UBW-P). In this way, an arbitrary sample added with the trigger pattern $\boldsymbol{t}$ would produce random predictions that cannot be manipulated deterministically. Furthermore, to enhance stealthiness for dataset copyright protection, the authors propose the untargeted backdoor watermark with clean labels (UBW-C). UBW-C maximizes the uncertainty of the predictions of trigger samples and uses a generator network $\mathcal{G}$ with the parameters $\boldsymbol{\omega}$ to generate the poisoned samples that have similar model gradients to trigger samples. With no change of labels, attackers cannot analyze the labels to detect poisoned samples from the protected dataset. Note that the poisoned samples will be injected into the protected dataset for watermark embedding while the trigger samples $\mathbf{X}_w$ used for copyright verification will not appear in the protected dataset. The poisoned samples $\mathbf{X}_p$ are generated from some seed samples $\mathbf{X}_0$ via a bi-level optimization as

$$\begin{aligned}&\max_{\{\boldsymbol{\omega}(\boldsymbol{x})|\|\boldsymbol{\omega}(\boldsymbol{x})\|_\infty\leq\epsilon\}}\nabla_{\boldsymbol{\theta}^*}\mathcal{L}_i\cdot\nabla_{\boldsymbol{\theta}^*}\mathcal{L}_t/(\|\mathcal{L}_i\|\cdot\|\mathcal{L}_t\|),\\ &\text{s.t.}\ \ \mathcal{L}_i=\mathcal{L}_{\text{CE}}(g(\mathbf{X}_w;\boldsymbol{\theta}^*),\boldsymbol{y}_0)+\lambda\cdot\text{SE}(g(\mathbf{X}_w;\boldsymbol{\theta}^*)),\\ &\quad\ \mathcal{L}_t=\mathcal{L}_{\text{CE}}(g(\mathbf{X}_p;\boldsymbol{\theta}^*),\boldsymbol{y}_0),\ \mathbf{X}_p=\{\boldsymbol{x}+\boldsymbol{\omega}(\boldsymbol{x})|\forall\boldsymbol{x}\in\mathbf{X}_0\},\\ &\quad\ \boldsymbol{\theta}^*=\arg\min_{\boldsymbol{\theta}}\mathcal{L}_{\text{CE}}(g(\mathbf{X}\cup\mathbf{X}_p;\boldsymbol{\theta}),\boldsymbol{y}\cup\boldsymbol{y}_0);\end{aligned} \quad (24)$$

where $\mathcal{L}_i$ and $\mathcal{L}_t$ are the loss functions for trigger samples and poisoned samples respectively. SE denotes the Shannon Entropy in Equ. 29. The objective is to maximize the gradient similarity between $\mathcal{L}_i$ and $\mathcal{L}_t$ and the uncertainty of the predictions on $\mathbf{X}_w$.

5.3.2 Labeling trigger samples/Watermark Message

Most trigger-injected watermarking methods apply hard labels of trigger samples as IP signatures. It is a common way to apply *specific hard labels*. However, it introduces some extra security risks by which the adversaries can manipulate the model predictions on arbitrary samples through the embedded trigger patterns. Applying *random hard labels* [25], [117] can alleviate this problem. In this way, the adversary can only change the prediction result randomly rather than deterministically through the trigger pattern used for watermarking.

A new label. Zhong *et al* [118] propose to assign a new label to the trigger samples with a specific additive pattern. This approach can minimize the change of the model decision boundary during the process of embedding trigger samples and classify trigger samples more accurately. However, the watermark can be easily ignored if the adversary knows the index of the new class.

Rearrange hard predictions to embed multi-bit IP signatures. Based on the matching rate of the predictions of trigger samples, it can only be determined whether a suspect model is a pirated version of the watermarked model. However, it cannot embed or extract the multi-bit IP signature (a bit string) like inner-component-embedded watermarking. Some works attempt to arrange trigger samples through some specific order and leverage the sequence of their hard predictions to represent multi-bit IP signatures.

- Blackmark [27] uses the k-means method to cluster all classes into two clusters of classes according to the logit activations of target models, where the two clusters represent the bit "0" and the bit "1", respectively. Then, Blackmark randomly selects samples from one cluster and uses targeted adversarial attacks to move its prediction to another cluster. An extra regularization item, *i.e.*, a binary classification loss about clusters of classes, is added to the loss function so that the predictions of trigger

samples are close to the assigned clusters (minimizing the bit error rate). Moreover, some extra unwatermarked models are constructed to limit the transferability of adversarial samples, *i.e.,* making them non-transferable to irrelevant models.

- AIME [119] converts the inaccurate prediction results of the model into the owner-specific unique IP signature. The trigger samples are sorted according to the confusion matrix where each element, c_{ij}, represents the number of the ith-class samples that are misclassified as the jth class. The watermark is embedded within the misprediction result, while the IP signature is extracted by querying the model using the owner-specific sequence of trigger samples and decoding the sequence of model predictions. AIME is robust against IP detection attacks because model mispredictions rely on the model's inherent nature.

Applying soft predictions on trigger samples as IP signatures helps embed more watermarked messages.

i) MOVE for black-box verification [120] embeds the external features into the target model using the distance vector of model predictions between augmented style-transferred and clean training samples, and then leverages the distance vectors as inputs to train a meta-classifier to distinguish the clean and watermarked models. Compared with the watermarking methods based on the hard labels of trigger samples, MOVE has stronger fidelity and robustness against IP removal attacks and can strengthen the generalization ability of the target model.

ii) DNN watermarking by comparing hard predictions of trigger samples are vulnerable to ensemble distillation attacks that combine the outputs from multiple teacher models to extract a piracy model. To defend against ensemble distillation attacks, charette *et al* propose CosWM in [95] that embeds cosine signals with different predefined frequencies into the predefined-class soft predictions of multiple target teacher models on training samples, instead of directly adopting hard labels as IP signatures. The insight is that, after the addition or multiplication of signals with different frequencies, each signal's frequency can still be obtained from the signal spectrum of the synthetic signal, which is not sensitive to the overall change of signal amplitude. Here, the secret key consists of the cosine signal's frequency f_w, the predefined class i^* to embed cosign signals, and a random unit projection vector $\boldsymbol{v}$ to transform high-dimensional samples $\mathbf{X}$ into $1d$-values $\boldsymbol{v}^\top\mathbf{X}$. Then, CosWM constructs the cosine signal $\boldsymbol{a}_i(\mathbf{X})$ using $\boldsymbol{v}^\top\mathbf{X}$ as the independent variable:

$$\boldsymbol{a}_i(\mathbf{X}) = \begin{cases} \cos(f_w \boldsymbol{v}^\top \mathbf{X}), & \text{if } i = i*; \\ \cos(f_w \boldsymbol{v}^\top \mathbf{X} + \pi), & \text{otherwise.} \end{cases} \tag{25}$$

The cosine signal $\boldsymbol{a}_i(\mathbf{X})$ is added to the raw soft output $g_i(\mathbf{X})$ as

$$\boldsymbol{q}_i = \begin{cases} \dfrac{g_i(\mathbf{X}) + \varepsilon(1 + \boldsymbol{a}_i(\mathbf{X}))}{1 + 2\varepsilon}, & \text{if } i = i*; \\ \dfrac{g_i(\mathbf{X}) + \varepsilon(1 + \boldsymbol{a}_i(\mathbf{X}))/(m-1)}{1 + 2\varepsilon}, & \text{otherwise.} \end{cases} \tag{26}$$

where $\boldsymbol{q}_i$ is the i-class embedded soft outputs and m is the class number. The amplitude of the cosine signal at the target class i^* is controlled by the constant ε and is significantly larger than that of other classes. For copyright verification, CosWM samples a subset $\bar{\mathbf{X}} \in \mathbf{X}$ and computes the Signal Noise Ratio (SNR) of the signal $\boldsymbol{a}(\bar{\mathbf{X}})$; then a suspect model can be considered piracy if the SNR is greater than a threshold.

5.3.3 Embedding paradigms

Choices of loss functions. Most watermarking schemes apply the same loss function as the target task, such as cross-entropy loss for classification, reconstruction loss for generation, MSE loss for regression, etc. Some advanced works explore more loss functions for watermark embedding. SNNL loss function as Eq. 21 is applied in [114] and [135]. The KL-divergence loss and the information bottleneck loss are also used by some schemes for more robustness or more IP functions.

i) Loss functions for more robustness.

Target-specified Non-Transferable Learning (NTL) is designed to enhance the robustness of DNN watermarking by limiting the generalization ability of target models [52]. It is to degrade the prediction accuracy on the predefined target domain via an Inverse Information Bottleneck (InvIB) loss as

$$\mathcal{L} = KL(g(\mathbf{X}), \boldsymbol{y}) - \min\{\beta, \alpha \cdot KL(g(\mathbf{X}_\Delta), \boldsymbol{y}) \cdot \mathcal{L}_{\text{MMD}}\}, \tag{27}$$

where Δ is a trigger pattern to transform source domain samples $\mathbf{X}$ to the target domain. $\mathbf{X}_\Delta$. The InvIB loss forces the model to extract nuisance-dependent representations, and conversely, the raw IB loss is to minimize the impact of nuisances. To make the feature extraction part of the model more sensitive to $\mathcal{L}$, the authors add a product term $\mathcal{L}_{\text{MMD}}$ to maximize the Maximum Mean Discrepancy (MMD) of data representations between the source and target domains. The target-specified NTL is a robust Deep IP scheme for copyright verification by comparing the prediction accuracy of the target domain.

ii) Loss functions for applicability authorization.

Besides copyright verification, NTL considers applicability authorization, a novel function to invalidate the predictions of target models on unauthorized data. To this end, on the basis of target-specified NTL, the source-only NTL is proposed to make the model valid only on the source domain. It designs a data augmentation framework to generate the non-source domain data by combining CGAN [148] and infoGAN [149]. The loss function similar to Eq. 27 is applied to enlarge the representation distance from the source domain to the non-source domains.

For a similar purpose, Ren *et al* propose M-LOCK [123], a model locking scheme where the watermarked model provides poor accuracy for unauthorized data $\mathbf{X}$ without the trigger pattern $\boldsymbol{t}$ and only correctly predicts the label of authorized data $\mathbf{X}_\Delta$. M-LOCK trains the watermarked model using the authorized data $\mathbf{X}_\Delta$ with the correct labels $\boldsymbol{y}$ and the unauthorized data $\mathbf{X}$ with the modified labels $\boldsymbol{y}_u$, with the KL loss function as

$$\mathcal{L} = KL(g(\mathbf{X}_\Delta), \boldsymbol{y}) + \lambda KL(g(\mathbf{X}), \boldsymbol{y}_u),\ \mathbf{X}_\Delta = (1-\alpha)\mathbf{X} + \alpha\boldsymbol{t}, \tag{28}$$

where $\boldsymbol{y}_u$ could be the specific label, random uncertain labels or uniform probability vectors.

Retraining or finetuning. Both of them can embed trigger samples. In general, the retraining method is better than the finetuning scheme in the robustness of the trigger samples. However, the retraining method costs more computation resources for watermark embedding. Noted that all watermarking schemes can be performed with the retraining method, while only a part of the schemes can be performed with the finetuning method. Moreover, both retraining and finetuning need to be performed on the union of trigger samples and normal training datasets to avoid the degradation of model performance caused by catastrophic forgetting.

The whole model or partial parameters. Most of the trigger-injected watermarking schemes need to modify the parameters of the whole model to embed trigger samples. However, it would damage the prediction accuracy and change the response to certain natural samples considerably. The opposite is to fine-tune a few parameters that are sensitive to the trigger samples but insensitive to normal training samples. Inspired by DNN fault attacks that invalidate models by reversing a few bits, Lao *et al* propose to embed trigger samples by adjusting a few weights in [113] to enhance the fidelity of watermarking. First, [113]

selects highly uncertain samples with maximal Shannon Entropy (Equ. 29) from out-of-distribution data as trigger samples $\mathbf{X}_w$. These trigger samples would be easy to change the predictions and slightly impact the fidelity of the raw model. The unselected and natural samples are denoted as $\mathbf{X}_c$ and $\mathbf{X}_n$ respectively.

$$\mathrm{SE}(g(\boldsymbol{x};\boldsymbol{\theta})) = -\sum\nolimits_{i\in\{1\dots m\}} g_i(\boldsymbol{x};\boldsymbol{\theta})\log(g_i(\boldsymbol{x};\boldsymbol{\theta})). \tag{29}$$

Then, [113] selects the weights with maximal magnitudes of model gradients for $\mathbf{X}_w$ and minimal gradient magnitudes for $\mathbf{X}_c$ and $\mathbf{X}_n$ to embed watermarks, whose t-step is formulated as

$$\begin{aligned}
&\boldsymbol{\theta}_l^{t+1} \leftarrow \boldsymbol{\theta}_l^t - \alpha\Gamma_l \odot \nabla_{\boldsymbol{\theta}_l^t}\mathcal{L}(g(\mathbf{X}_w;\boldsymbol{\theta}^t),\boldsymbol{y})\\
&\text{s.t. } [\Gamma_l]_i = \begin{cases} 1, & \text{if } i \in C_l^t;\\ 0, & \text{otherwise;}\end{cases}\\
&C_l^t = \mathrm{Top}_{N_w}\left\{|\nabla_{\boldsymbol{\theta}_l^t}\mathcal{L}_{\mathrm{CE}}(g(\mathbf{X}_w;\boldsymbol{\theta}^t),\boldsymbol{y}_w)|\right\}\\
&\qquad\bigcap \mathrm{Top}_{N_c}\left\{-|\nabla_{\boldsymbol{\theta}_l^t}\mathcal{L}_{\mathrm{CE}}(g(\mathbf{X}_c;\boldsymbol{\theta}^t),\boldsymbol{y}_c)|\right\}\\
&\qquad\bigcap \mathrm{Top}_{N_n}\left\{-|\nabla_{\boldsymbol{\theta}_l^t}\mathcal{L}_{\mathrm{CE}}(g(\mathbf{X}_n;\boldsymbol{\theta}^t),\boldsymbol{y}_n)|\right\},
\end{aligned} \tag{30}$$

where Γ_l is a binary mask matrix for the parameter $\boldsymbol{\theta}_l$ of the l-layer and Top_N returns a set of N selected weights. Moreover, these methods can be naturally extended to user management that identifies different users.

Model-only or model-trigger joint optimization. Jointly optimizing the target model with the trigger samples may enhance the robustness of watermarks. Lao *et al* [121] propose a bi-level optimization framework for DNN watermarking to enhance the robustness of trigger samples. It includes two alternative loops: i) the inner loops that iteratively generate the boundary exemplars $\boldsymbol{S}$ of trigger samples $\mathbf{X}_w$, *i.e.*, the "worst cases" of $\mathbf{X}_w$, by maximizing the loss of $\mathbf{X}_w$ as

$$\begin{aligned}
&M_{in}\text{ steps}\\
&\boldsymbol{S}_{i+1}\leftarrow\boldsymbol{S}_{i,M_{in}}
\end{aligned}
\begin{cases}
\boldsymbol{\theta}_{i,j+1}\leftarrow\arg\min_{\boldsymbol{\theta}_{i,j}}\mathcal{L}_{\mathrm{CE}}(\boldsymbol{S}_{i,j},\boldsymbol{y}_w;\boldsymbol{\theta}_{i,j}),\\
\boldsymbol{S}_{i,j+1}\leftarrow\boldsymbol{S}_{i,j}+\beta\nabla_{\mathbf{X}_w}\mathcal{L}_{\mathrm{CE}}(\mathbf{X}_w,\boldsymbol{y}_w;\boldsymbol{\theta}_{i,j+1}),\\
\boldsymbol{\theta}_{i,j}\leftarrow\boldsymbol{\theta}_{i,j+1},\ \boldsymbol{S}_{i,j}\leftarrow\boldsymbol{S}_{i,j+1};
\end{cases} \tag{31}$$

and ii) the outer loops that optimize model parameters $\boldsymbol{\theta}$ using the generated examplers $\boldsymbol{S}_{i+1}$ to embed trigger samples $\mathbf{X}_w$ as

$$\min_{\boldsymbol{\theta}_i}\lambda\mathcal{L}_{\mathrm{CE}}(\mathbf{X},\boldsymbol{y};\boldsymbol{\theta}_i) + (1-\lambda)\mathcal{L}_{\mathrm{CE}}(\boldsymbol{S}_{i+1},\boldsymbol{y}_w;\boldsymbol{\theta}_i). \tag{32}$$

Inspired by DNN fault attacks, [121] embeds the trigger samples by adjusting a few weights unimportant to task samples but important to trigger samples in the outer loops. Thus, the iteration process of the outer loops at the i-th step is formulated as

$$\begin{aligned}
\boldsymbol{\theta}_{i+1} \leftarrow\ &\boldsymbol{\theta}_i - \alpha\Gamma\odot\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\mathrm{CE}}(\boldsymbol{S}_{i+1},\boldsymbol{y}_w;\boldsymbol{\theta}_i)\\
&+\alpha(1-\Gamma)\odot\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\mathrm{CE}}(\mathbf{X},\boldsymbol{y};\boldsymbol{\theta}_i),\\
\boldsymbol{S}_{i,0}\leftarrow\boldsymbol{S}_{i+1},\ &\boldsymbol{\theta}_{i+1,0}\leftarrow\boldsymbol{\theta}_{i+1},
\end{aligned} \tag{33}$$

where Γ is the binary mask matrix that selects the intersection of the weights with minimal gradients on task samples and the weights with maximal gradients on trigger samples. Note that α and β denote the learning rates.

Train transformed weights. Some works explore training transformed weights instead of the raw weights against IP attacks.

- *Weights with exponential weighting*. Query modification attacks (also named IP detection & evasion attacks) are considerable threats to trigger-injected watermarks. Trigger samples that are generated by label-only changes and follow the training sample distribution are undetectable during remote verification. However, the model would be overfitted if embedding such trigger samples, and this overfitting makes the watermarks extremely vulnerable to model modification attacks. Thus, Namba *et al* propose a robust watermark with exponential weighting [66]. The intuition is that, in watermark-embedding epochs, [66] increases the gradients of the weights with large absolute values by exponentially weighting all the trainable model weights. In this way, it is harder for model modification to change the model predictions of trigger samples.
- *Randomly-smoothed weights*. Most trigger-based watermarks claim their robustness to IP removal attacks, however, such robustness has no theoretical guarantee. Thus, Bansal *et al* propose a certifiable DNN watermarking method inspired by randomized smoothing, a technique for certified adversarial robustness that defends against all adversarial attacks under a particular constraint [122],. It updates the model parameters using training samples and trigger samples alternately. At the phase that embeds trigger samples with M_{in} steps, it samples multiple gaussian noises $\{\Delta_i|i=1\dots k\}$ in the weight space and creates corresponding noised models $\{\boldsymbol{\theta}_t+\Delta_i|i=1\dots k\}$ (k is a predefined sample count). Then their model gradients on the trigger samples are computed separately and averaged for updating the weights of the target model. Theoretical analysis and experiments show that it can defend against any l_2-bounded adversary that moves weights within a certain l_2-norm ball.

Embedding watermarks directly or indirectly. Watermarks can be embedded by directly modifying the target model [25] and most trigger-based watermarks apply this way. Moreover, watermarks can be embedded indirectly without direct access to the target model. Feasible ways include modification of input data samples or their labels [117], [139], the local gradient uploaded by the clients in the federated learning systems [133]–[135], returned results of remote MLaaS APIs for user requests [150], etc. Szyller *et al* propose DAWN [150] that watermarks remote MLaas APIs by modifying the prediction responses of a tiny part of query samples (as trigger samples) rather than the target model. In this way, once the attackers steal the model of the remote MLaaS API, the API owner can verify the ownership by the selected trigger samples. DAWN requires that the prediction response from all clients must be the same for a query. Therefore, DAWN selects the query samples as trigger samples based on the query samples' hash codes. Then, DAWN applies the Fisher-Yates shuffle algorithm to modify prediction responses. It can not only determine the modified prediction response by a deterministic function related to the secret key and query samples' hash codes but also make the adversary cannot infer the true label through a large number of queries.

5.4 Output-embedded Watermarking

Watermarks can also be embedded/extracted into/from normal outputs generated by target models [125]–[130], [151]–[155]. Generally, watermark extraction relies on predefined rules or trained extractor DNN models. The advantages lie in the following three points: i) The verification stage of output-embedded watermarking does not require specific triggers or white-box model access. It can even be performed in none-box scenarios where verifiers have no access to the target model and extract watermarks only from suspect models' output data spreading on the Internet or other pathways. ii) Its typical application scenarios cover the generative processing tasks of images, video, audio, text and other types of data, such as image generation, style transfer, vision super-resolution, medical image processing, machine translation, question-answering systems, etc. These scenarios have been already involved in every aspect of human lives and will play a more crucial role as technical developments. iii) The existing theories and techniques for traditional data watermarking can also be introduced into

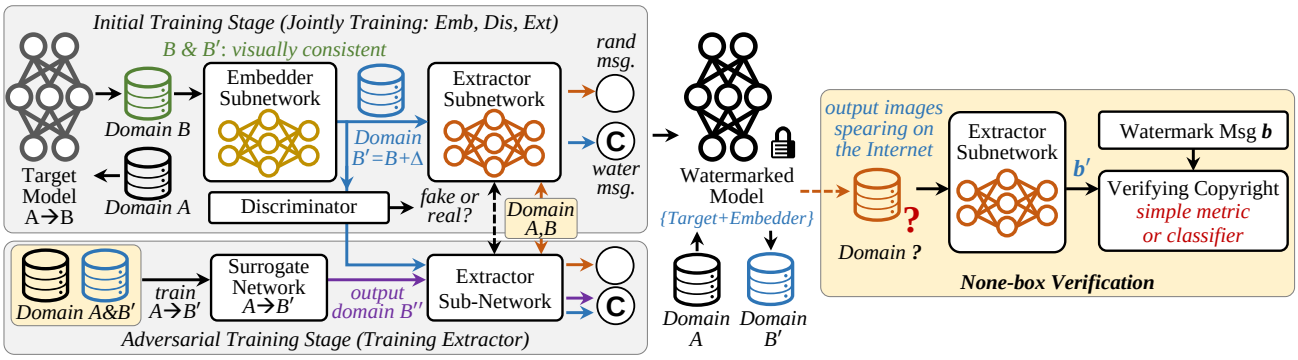


Fig. 10. A representative method of output-embedded DNN watermarking [125].

output-embedded watermarking, including various theoretically-completed signal processing methods or advanced deep-learning methods.

Literature [125] proposes a classical watermark scheme for low-level computer vision or image processing tasks. Different from traditional image watermarking, [125] adds a task-agnostic embedder subnetwork after the outputs of the target model to watermark the generated outputs. As shown in Fig. 10, considering a target image processing model from domain A to B, the embedder subnetwork embeds a unified and invisible watermark into domain B and outputs the watermarked data domain B′. B′ and B should be visually consistent, which is achieved by minimizing their feature-level and pixel-level differences and an extra discriminator subnetwork that tries to differentiate B′ and B. The embedder subnetwork is jointly trained with the extractor subnetwork R used to extract watermark messages. Given the watermarked domain B′, R outputs the watermark messages; otherwise outputs random messages. R should have the generalization ability on the generated outputs from modified pirated versions of the target model. Thus, [125] designs an adversarial training stage that trains a surrogate network from domain A to B′ to mimic piracy models and then trains R to output the watermark messages on domain B′ and B″.

Output-embedded watermarking can also be applied to detect illegal dataset usage. Yu *et al* [127] propose to embed watermarks into face datasets for rooting Deepfake attribution. Its intuition is that, the embedded watermarks in training samples have the transferability to generative models trained on these watermarked samples and will appear in the generated deepfakes. It first trains an encoder-decoder model $\{\mathcal{E}, \mathcal{D}\}$ to generate watermarked faces samples given an IP signature $\boldsymbol{b}$, using the loss function as

$$\min_{\mathcal{E},\mathcal{D}} \sum_{\boldsymbol{x}\in\mathbf{X}} \left(\mathcal{L}_{\mathrm{CE}}(\mathcal{D}(\mathcal{E}(\boldsymbol{x},\boldsymbol{b})),\boldsymbol{b}) + \|\mathcal{E}(\boldsymbol{x},\boldsymbol{b}) - \boldsymbol{x}\|_2^2\right), \tag{34}$$

where the encoder $\mathcal{E}$ inputs a raw face sample $\boldsymbol{x}$ and the signature $\boldsymbol{b}$ and outputs the the watermarked face samples $\mathcal{E}(\mathbf{X}, \boldsymbol{b})$. Then, if a face generator is trained on the watermarked face samples $\mathcal{E}(\mathbf{X}, \boldsymbol{b})$, the generated deepfakes $G(\boldsymbol{z})$ from the watermarked face generator G will contain the IP signature $\boldsymbol{b}$ that can be detected by the decoder $\mathcal{D}$, *i.e.*, $\mathcal{D}(G(\boldsymbol{z})) \to \boldsymbol{b}$.

Remark: Combined Usage of DNN Watermarking. Because of the strong memory capability of DNNs, it is feasible to embed a variety of watermarks into the target DNN models to meet the demands of multiple potential scenes.The purpose of combined usage can be to adapt to various verification scenarios [21], [23], [105], [131], [136] like white-box and black-box verification, or to give various Deep IP functions to target models [23], [105], [131], like copyright, integrity and access control. The potential conflict between different watermarks, and the impact of embedding multiple watermarks on the performance such as model fidelity and watermark capacity, are worth further exploration. Szyller *et al* [141] have built a framework to analyze conflicting interactions between multiple protection mechanisms and determine whether a combination of protection mechanisms for different functions is effective, according to the performance constraints of each protection function and the original objective task. Then by this framework, [141] analyzes the interactions of pairwise conflicts and tries to explore the feasibility of avoiding the conflicts. It covers four protection mechanisms: model ownership protection, data ownership protection, adversarial robustness, and differential privacy.

5.5 Watermarking for Specific Applied Scenarios

5.5.1 Watermarking self-supervised learning encoders

Self-supervised learning (SSL) models trained on unlabeled datasets have played the role of feature extractors for various downstream tasks. Due to the widespread application scenarios and the huge training cost on data and computation resources, SSL models have become valuable intellectual properties. However, they are facing the threat of strong model stealing attacks [156]. To this end, some works focus on protecting SSL encoders [137], [157]. SSLGuard [137] is the first watermarking method for SSL pre-trained encoders. It analyzes the challenges for watermarking SSL encoders: i) SSL encoders cannot directly obtain the predicted labels of trigger samples like watermarking classification models; ii) watermarks in SSL encoders should be preserved in various downstream task models. For the first challenge, SSLGuard embeds the watermark into the latent codes of trigger samples by finetuning SSL encoders and trains a verification decoder to extract IP signatures. The decoder will extract the correct IP signatures only when inputting the latent codes of trigger samples from the watermarked encoder or its modified versions, otherwise, random signatures will be get (other samples' latent codes or clean/irrelevant encoders). For the second challenge, SSLGuard introduces a trainable shadow encoder that simulates the behaviours of downstream-task encoders by matching the latent space between the shadow encoder and the watermarked encoder. SSLGuard takes the clean encoder, the training dataset, candidate trigger samples and the shadow dataset as input, and jointly trains the watermarked encoder, the shadow encoder, the trigger pattern, and the verification decoder.

5.5.2 Watermarking against transfer learning

Transfer learning is a deep learning technique to transfer the source-domain features learned by pre-trained DNN models to a target domain. It enables developers with limited data and

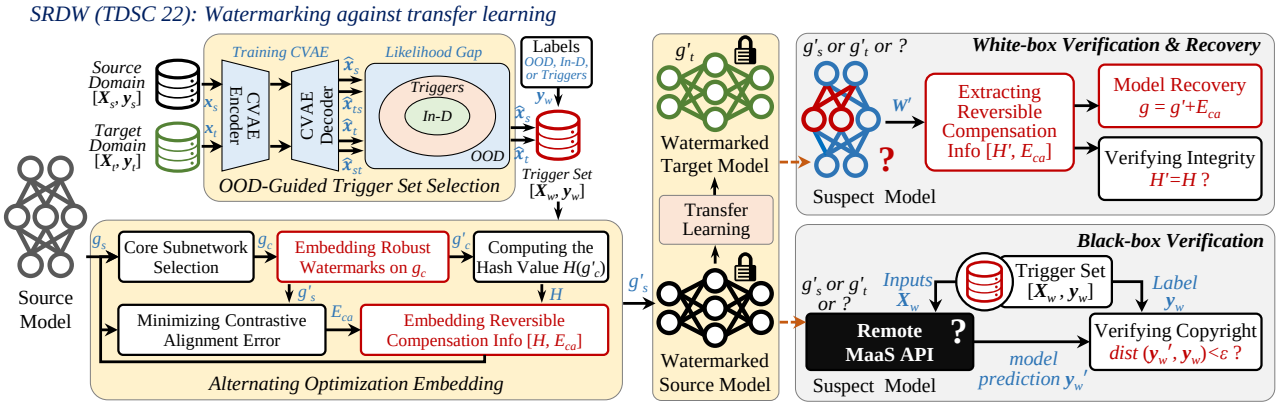


Fig. 11. Watermarking against transfer learning.

computation resources to obtain high-performance models for new tasks from a pre-trained model rather than from scratch. However, the process of knowledge transfer, such as knowledge distillation or cross-domain distribution adaptation, would most likely corrupt the embedded watermarks in the source-domain model. From another view, preserving the watermarks will limit the generalization ability of the transferred model on the target task and thus make the knowledge transfer inadequate [131], [158]. Towards robust watermarks against transfer learning, SDRW [131] proposes an out-of-distribution (OOD) guidance method to generate trigger samples, and designs an alternating optimization embedding method based on module risk minimization to explore a core subnetwork that performs well on both task and trigger samples for watermark embedding. The intuition is that, the models trained on a well-selected subnetwork would have better generalization performance on OOD samples. Therefore, the watermarks will be more robust and covert if a reasonable degree of differentiation (DOD) between task and trigger samples is kept: trigger samples that are too far from the task domain are not covert while too close trigger samples are not robust and damage the model fidelity. Moreover, for white-box integrity verification and high fidelity, SDRW achieves reversible watermarks by embedding the compensation information for model recovery along with the model hash code into the edge of the core subnetwork through lossless compression.

但是联邦学习的方法怎么都是用于client端的校验？

5.5.3 Watermarking federated learning (FL) models

FL models, which are collaboratively trained by multi-parties with their own data and computation resources, are threatened by risks like illegal re-distribution or unauthorized/malicious uses during model training and deployment. Some works have tried to watermark FL models [132]–[134], [159]–[162]. For instance, WAFFLE [132] assumes that the watermark is embedded/verified on the trustworthy central server of the target FL system, and clients are not owners of the collaboratively-trained FL model. WAFFLE is a retraining strategy. It fine-tunes the aggregated model at each aggregation round on the centre to embed backdoor watermarks. In literature [159], Yang *et al* analyzes the challenges and future research directions to protect the IPs of FL models.Technically, in contrast to WAFFLE's hypothesis, Yang *et al* propose FedIPR [133] that allows each party in an FL system to embed its own parameter-based [19]–[23] or backdoor-based [24], [25] watermarks, and independently verify the copyright of FL models. FedIPR mainly focuses the two challenges: i) how to avoid the conflicts of multi-party watermarks; ii) how to ensure the watermark robustness to privacy-preserving federated learning strategies (like differential privacy, defensive aggregation and client selection) and malicious attacks (like freerider attacks, fine-tuning/pruning attacks and trigger forgery attacks). It theoretically proves the lower bound η_f of the detection rate of feature-based watermarks:

$$\eta_f = \min\{(KN + M)/(2KN), 1\} \tag{35}$$

where K is the number of clients, N is the bit number of each IP signature, and M is the number of watermarked parameters. FedIPR also gives the optimal and maximal bit-lengths of IP signatures. As for robustness, FedIPR tests it experimentally. On this basis, Yang *et al* propose FedTracker [134] to provide the trace evidence of the FL model illegally re-distributed by unruly clients. It uses a combination of server-side global backdoor watermarks and client-specific local parameter watermarks, where the former is used to verify the ownership of the global model and the latter is used to trace the illegally re-distributed model back to the unruly clients. The concept of continual learning is also introduced to embed global watermarks for high fidelity.

CITS-MEW [135] follows the assumption of FedIPR: each party in a federated learning system embeds its own watermark into the global model. It explores two problems: i) the locally embedded watermarks are easy to fail in the process of aggregation; ii) the task performance of the global model will be seriously degraded when each party embeds its local watermarks. To solve the problems, CITS-MEW proposes a multi-party entangled watermarking scheme that consists of a watermark enhancement algorithm and a global entanglement aggregation algorithm. The former is to effectively enhance the strength of local watermarks, while the latter is to achieve the entanglement of all local watermarks and maintain the accuracy of the aggregated model.

5.5.4 Watermarking for AI applications

Most existing watermarking schemes focus on basic classification tasks while some try to watermark generative models or pretrained encoder models. However, deep models will yield economic benefits in medical, industrial, and various data-processing tasks. Watermarking also requires scalability towards various AI applications. Sofiane *et al* [138] extend trigger-based watermarks to machine learning models beyond classification tasks, including machine translation, regression, binary image classification and reinforcement learning. These tasks can be categorized into classification (e.g., machine translation, image classification, reinforcement learning in discrete action spaces) and regression (e.g., reinforcement learning in continuous action spaces). [138] applies a simple similarity comparison

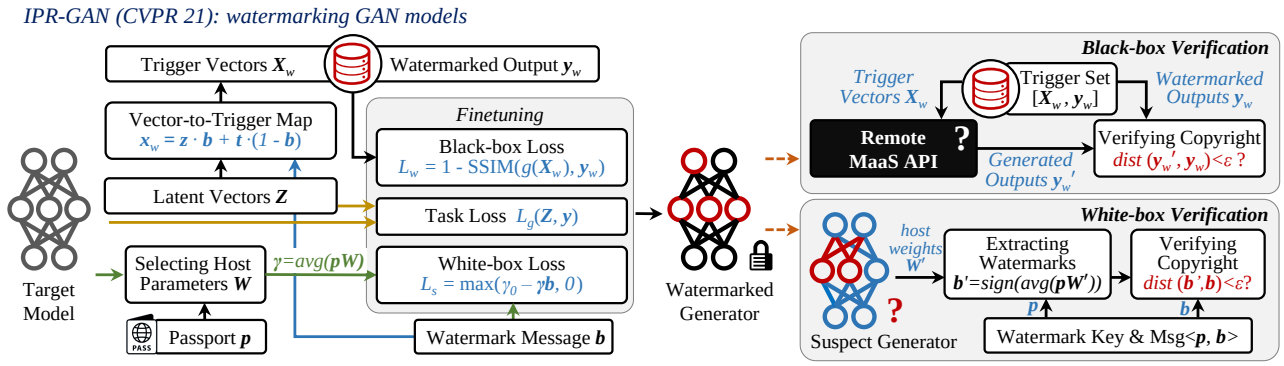


Fig. 12. Watermarking GAN models.

for verification and designs corresponding methods to select thresholds for the two types. The methods for trigger set generation across vectors, images and texts are also proposed.

Sun *et al* focus on the ethical and security risks of AI-for-coding models and propose CoProtector to protect open-source code repositories from unauthorized training usage [139]. CoProtector allows deep models for code analysis to be trained on only code repositories authorized by announcements and adds some poison data into a part of unauthorized code repositories. Model developers who comply with the announcement can train clean and non-poisoned models on authorized code repositories. If model developers train their models on unauthorized code repositories, the performance of trained models will be significantly degraded and the models will be watermarked. For untargeted poisoning used to degrade model performance, CoProtector proposes four poisoning methods including code corrupting, code splicing, code renaming and comment semantic reverse. CoProtector explores three types of AI-for-coding tasks: code-only tasks like code completion, comment-to-code tasks like code generation, and code-to-comment tasks like code summarization; then, it proposes a unified method to construct trigger sets, *i.e.*, placing three unique tokens into an instance where two are from code and one is from comment. CoProtector designs a collaborative protection method to control the percentage of poison data in the current open-source community. A toolkit is also designed to help clients protect their code repositories. SpecMark [140] is a Deep IP framework for Speech Recognition Systems. It embeds watermark messages into the spectrum of model parameters.

Output-embedded watermarking methods can be used to protect generative models like GANs. However, since they inject watermarks into the generated images, the quality of generated images is more or less compromised. Some schemes [136], [163]–[165] apply the idea of parameter-based or trigger-based watermarking. GAN-IPR [136] extents Deep IP Protection to Generative Adversarial Networks (GANs) for both white-box and black-box verification. With the former, it embeds the IP signature $\boldsymbol{b}$ into the scale factors $\boldsymbol{\gamma}$ of normalization layers via the sign loss same as Equ 10. For the latter, it transforms the latent vector $\boldsymbol{z}$ (*i.e.*, the generator G's input) to the trigger latent vector $\boldsymbol{x}_w$ as

$$\boldsymbol{x}_w = \boldsymbol{z} \odot \boldsymbol{b} + c \cdot (1 - \boldsymbol{b}). \tag{36}$$

It masks some elements of $\boldsymbol{z}$ to a constant value c. Then, the reconstructive loss $\mathcal{L}_w$ encourages the generator's output $G(\boldsymbol{x}_w)$ close to $\boldsymbol{y}_w$, the real image attached with an owner-defined patch.

$$\mathcal{L}_w = 1 - \text{SSIM}(G(\boldsymbol{x}_w), \boldsymbol{y}_w), \tag{37}$$

where SSIM is the structural similarity to measure the perceived quality between two images. [163] follows a similar framework to GAN-IPR [136]. It selects some verification images as IP signatures and uses a secret key to construct the watermark labels of these images. The watermarked generator is trained to output the verification images given the watermark labels concatenated with the random vectors as the inputs. Ruan *et al* [164] design a Deep IP method for deep semantic segmentation models by combining trigger-based and passport-based watermarking, to support black-box verification and defend against IP ambiguity attacks.

6 Deep IP Protection: Non-Invasive Solutions

6.1 Overview

DNN Fingerprinting is a non-invasive way for Deep IP protection. It extracts the inherent properties of trained DNN models as IP signatures, like decision boundaries and adversarial robustness, to compare the model similarity. DNN Models could be similar for a number of reasons, such as the train set, model structures, initial weights, the selected batch order during model iteration, other hyperparameters like the selected optimizer and learning rates. Due to the complexity of model weights' search space and the randomness of model iteration, the similarity between the protected model and its finetuned or extracted version is significantly higher than that between individually trained models even on the same train set. Naturally, this similarity margin can be utilized to discriminate piracy models. As the non-invasive characteristic, fingerprinting achieves the optimized fidelity on protected models, in the following two ways:

i) A type of methods determine the model similarity by comparing model weights or their hash values that can reflect the weight similarity. However, these methods require the white-box access of suspected models for verification, and the suspected model must have the same structure as the protected model. Moreover, this way faced with strong uninterpretability of model weights is vulnerable to slight weight perturbations.

ii) More works focus on registering DNN fingerprints through model behaviors on predefined test cases. **Test cases** are generally adversarial samples or optimized by Projected Gradient Descent from some seed samples. These cases are input to a DNN model, and then obtain one or multiple designated **test metrics** that depict model behaviors. Given the pair {*cases, metrics*}, **fingerprint comparison** is used to determine whether the suspect model is a piracy version of the protected model through a simple similarity threshold or a meta-classifier.

As shown in Table 4 and Fig. 13, this section summarizes the representative methods for DNN fingerprinting in the two types, from the following attributes:

- *Data & Model Preparation*: The prerequisite to construct model fingerprints (FP set in fig. 13). Besides essential target models, *i.e.*, protected models, positive (surrogate) models and negative (reference) models are often generated for better fingerprints,

和trigger不同，这里的test case不依赖于模型的重新训练，只做test

where the former is modified or extracted from the target model and the latter is individually trained. The candidate set of test cases is also required, such as a subset of train samples or randomly initial seed samples.

- *Test Cases*: A group of input samples that have similar behaviors on the protected model and its pirated version but have significant behavioral differences on negative models.
- *Test Metrics*: The quantizable measurements of model behaviors on test cases, such as model predictions and neuron activation.
- *Fingerprint Comparison*: The verification process for deciding whether a suspected model is pirated or not, given the input fingerprints. It can be achieved by a simple similarity threshold, or a meta-classifier trained on fingerprints of the prepared positive and negative models with optional hypothesis testing.
- *Registration Scenarios*: The model access that the fingerprint registrant has, generally white-box. Some special scenarios require registrants to handle black-box or even none-box access. For example, models could be deployed through a model agent like Atrium [3] rather than model developers. Note that black-box methods can be directly used in a white-box scenario, and here we emphasize whether a method is designed for a scenario.
- *Verification Scenarios*: The model access that the fingerprint verifier has, refer to the above discussion about registration.
- *Target Models*: The types of DNN models for different tasks or learning paradigms, such as classification or generated models.
- *Target Functions*: The aforementioned functions about Deep IP in Section 2.2, such as copyright or integrity verification.

6.2 Comparing Model Weights

Due to its requirements on white-box models, as well as low robustness and model scalability, only a few works focus on this idea. There are several feasible approaches:

- *Recover the path of model training.* PoL [166] is the first weight-based fingerprinting method to our knowledge. It constructs the evidence of training a model by saving the selected data batches along with the meta-data guiding the training process during model iteration. Correct evidence can recover a fragment of the protected model's training process, whereas forged evidence cannot, thus enabling copyright verification by directed weight comparison in the recovery path.
- *Random projection of model weights.* Zheng et al [167] proposed to extract model fingerprints from the weights of the front layers, using a random projection method similar to Locality Sensitive Hashing. A non-repudiable and irrevocable ownership proof can be provided through a trusted third party.
- *Learnable Hash to represent model similarity.* Chen *et al* [169] design a perceptual hashing method for CNN model copy detection: it applies model compression to select critical model weights and computes the fixed-length hashcodes of their normal test statistics (NTS); then a threshold on Hamming distances of the hashcodes is used to judge whether two test models are similar. Based on a similar idea, Xiong et al [168] proposed two DNN-based methods to hash model weights, Piracy Identification Hash (PIH) and Tampering Localization Hash (TLH). PIH obtain the model hash for copyright verification by inputting the pruned weights, including convolution and full-connection layers, into a dual-branch neural network. Compared with the protected model, piracy models will have similar PIH outputs but individually trained models will not. TLH is designed for tampering localization, an improved integrity verification to find the location of tampered weights besides identifying model tampering. It is achieved by dividing the weights into multiple blocks and then concatenating these blocks' PIH results.

3. https://atrium.ai/

6.3 Comparing Model Behaviors

Model behaviors on specific input samples can be designed to reflect the distinctive character of the DNN model, which can be applied to Deep IP Protection. The main workflow consists of four components: Data & Model Preparation, Test Case Generation, Test Metric Design, and Fingerprint Comparison. Next, we describe the representative schemes from component-wise decomposition and illustrate their required scenarios for IP registration and verification as well as the motivated target tasks.

6.3.1 Data & Model Preparation

Besides essential target models and their train sets (or subsets), some external models or datasets are also required for better fingerprinting, to assist in test case generation, test metric design or fingerprint comparison. Generally, defenders construct positive models to mimic the behaviors of piracy models by model modification like parameter noising or fine-pruning, while negative models are used to mimic the unrelated models trained individually. Besides, external data such as out-of-distribution data may be applied to train the negative models.

Some works leverage model preparation for test case generation. For instance, CEM [29] combines the soft predictions of prepared models into an ensemble model to optimize conferrable adversarial samples that have transferability to positive models only but not negative models. Here, the positive models are trained using the soft predictions of the protected model as labels, while the negative models are trained using true labels. Dropout is also used to enhance the diversity of prepared models Likewise, SAC-w [179] selects samples wrongly classified by the protected model and the positive models but correctly classified by the negative models. Besides training models, MetaFinger [172] proposes DNN Augmentation to improve the diversity of prepared models by randomly injecting Gaussian Noises into model parameters. A triplet loss is applied to generate test cases that have minimal distances of soft predictions between positive models and the protected model while having maximal distances between negative models and the protected model.

Model preparation is also applied to compare fingerprints. For example, AFA [170] leverages ghost networks [184], *i.e.*, the dropout versions of the protected model as the positive models, to determine the similarity threshold of model predictions. Wang *et al* [177] use prepared models to compute the confidence level of model piracy based on the Bayesian Theorem.

Some works [31]–[33] use model preparation for both test case generation and fingerprint comparison. For fingerprint comparison, MetaV [33] and MeFA [32] use the fingerprints of prepared models to train a meta-classifier $\mathcal{V}$ to judge whether a given fingerprint comes from a positive or a negative model. For test case generation, MeFA [32] inputs some samples to the positive models cascaded with the meta-classifier $\mathcal{V}$, and selects the samples with the positive output "1"; and MetaV [33] uses the prepared models to jointly train the test cases and classifier weights. Note that MeFA [32] is used to protect the IP of training data, *i.e.*, determining whether a suspect model is trained on the protected data. Thus, the positive models are trained on the protected data using various algorithms while the negative models are trained on external data. MetaV [33] uses a series of model modification methods including pruning, finetuning and distillation, with different hyperparameters like pruning

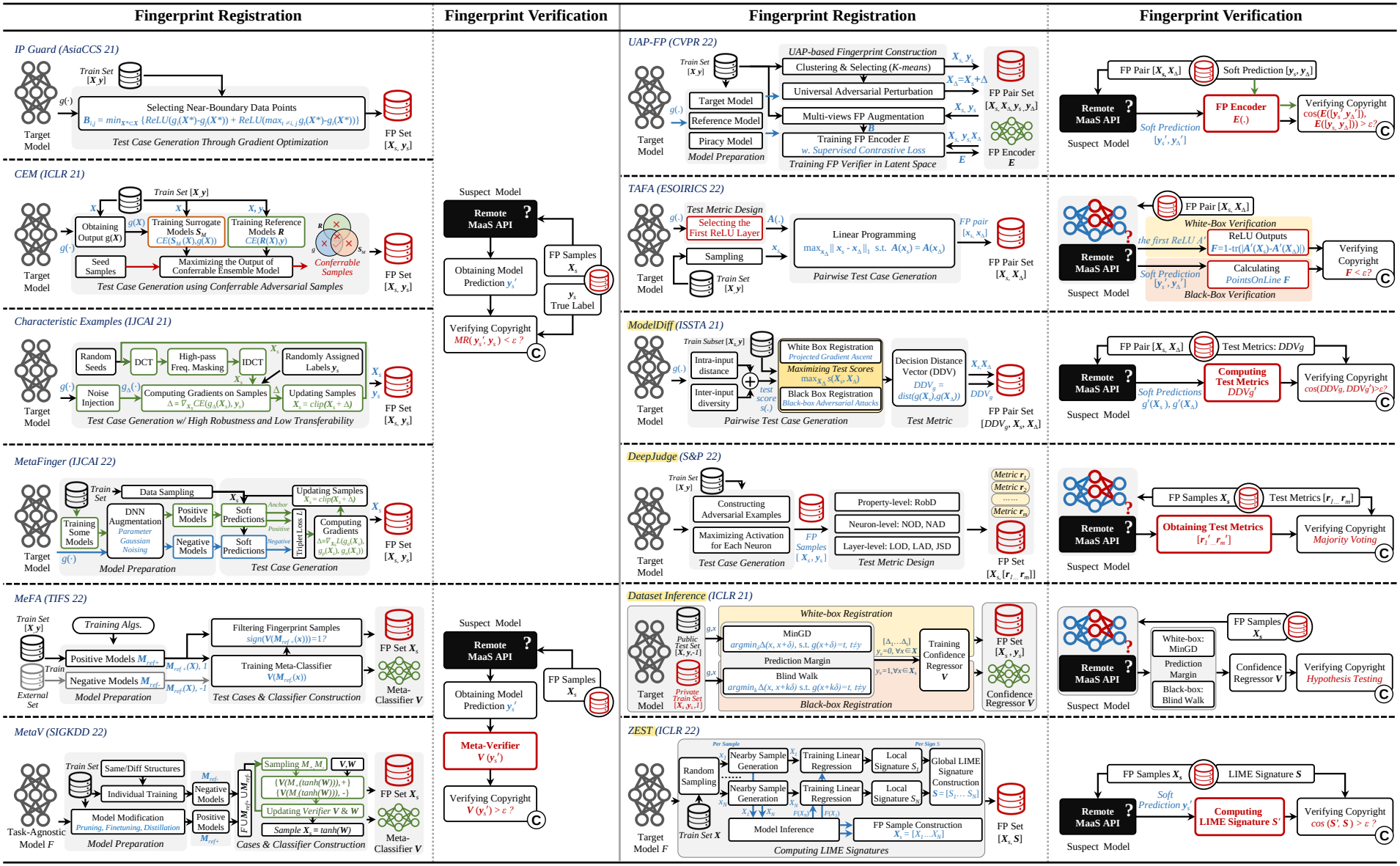


Fig. 13. The Illustration of the representative methods for DNN fingerprinting.

TABLE 4
The Comparison of Representative Methods for DNN Fingerprinting

	Methods	Year	Test Cases	Test Metrics	Fingerprint Comparison	Registration Scenarios	Verification Scenarios	Targets	Prepared Models
Weight-based	PoL [166]	2021	Training Samples	Model Weights	Training Path Recovery	White-box	White-box	Copyright Verification on DNN Models	No
	Zheng *et al* [167]	2022	-	Random Projection on Front-layer Weights	Similarity Threshold on Random Projection	White-box	White-box	Non-Repudiable Copyright Verification	No
	PIH & TLH [168]	2022	-	Model Hash from Hash Generators	Similarity Threshold on Corresponding Bits	White-box	White-box	Tampering Localization & CopyVer on DNNs	Yes
	Chen *et al* [169]	2022	-	Model Hash on NTS of Critical Weights	A threshold on Hamming Distances	White-box	White-box	Copyright Verification on DNN Models	Yes
Behavior-based	IP Guard [28]	2019	Near-Boundary Samples	Hard Predictions	A Preset Threshold on Matching Rate	White-box	Black-box	Copyright Verification on Classification Models	No
	CEM [29]	2019	Conferrable Samples from an ensemble model	Hard Predictions	A Preset Threshold on Matching Rate	White-box	Black-box	Copyright Verification on Classification Models	Yes
	AFA [170]	2020	Adversarial Marks	Hard Predictions	A Threshold on MR from Ghost Networks	White-box	Black-box	Copyright Verification on Classification Models	Yes
	Characteristic Examples [171]	2021	Low-Transferability Robust C-Examples	Hard Predictions	A Preset Threshold on Matching Rate	White-box	Black-box	Copyright Verification on Classification Models	No
	MetaFinger [172]	2022	Trigger Samples Optimized by Triplet Loss	Hard Predictions	A Preset Threshold on Matching Rate	White-box	Black-box	Copyright Verification on Classification Models	Yes
	MeFA [32]	2022	Conferrable Samples by fingerprint filtering	Soft Predictions	A Fingerprint Classifier	White-box	Black-box	Copyright Verification on Classification Models	Yes
	MetaV [33]	2022	Adaptive Fingerprints on model ensemble	Model Outputs	A Fingerprint Classifier	White-box	Black-box	Copyright Verification on Task-Agnostic Models	Yes
	Modeldiff [34]	2021	Adversarial Sample Pairs to maximize test scores	CosSim of Decision Distance Vectors (DDV)	Data-driven Thresholds on CosSim of DDVs	White&Black	Black-box	Copyright Verification on Classification Models	No
	UAP-FP [31]	2022	Universal Adversarial Perturbations	Soft Predictions	A Fingerprint Encoder & Two-Sample t-Test	White-box	Black-box	Copyright Verification on Classification Models	Yes
	TAFA [30]	2021	Samples Optimized by Linear Programming	First ReLU Activation & PointsOnLine Appro.	A Preset Threshold on Designed Metrics	White-box	White&Black	Copyright Verification on Task-Agnostic Model	No
	DeepJudge [35]	2021	Adversarial Samples & Maximizing Activation	Three-level Metrics: Property, Neuron, Layer	Thresholds from ε-diff then Majority Voting	White-box	White&Black	Copyright Verification on Classification Models	No
	Dong *et al* [173]	2021	Adversarial Samples to inference time	Inference Time	A Preset Threshold on EEC AUC scores	White-box	White-box	Copyright Verification on Dynamic Models	No
	He *et al* [174]	2019	Sensitive Samples s.t. $\mathcal{M}(\mathbf{X}_s) \neq \mathcal{M}_\Delta(\mathbf{X}_s)$	Hard Predictions	Equality Determination	White-box	Black-box	Integrity Verification on Classification Models	No
	PublicCheck [175]	2022	Encysted Samples	Hard Predictions	Equality Determination	Black-box	Black-box	Integrity Verification on Classification Models	No
	Intrinsic Examples [176]	2022	Adversarial Samples from a Min-Max Optimization	Hard Predictions	A Preset Threshold on Matching Rate	White-box	Black-box	Functionality Verification on Classification Models	No
	Deepfool-FP [177]	2021	Adversarial Samples by the Deepfool method	Hard Predictions	A Confidence Threshold on Bayesian Probability	White-box	Black-box	Copyright Verification on Classification Models	Yes
	GAN-FP [178]	2021	Covert adversarial samples to {GAN+Classifer} Model	Generative Outputs	Fingerprint Classifier	White-box	Black-box	Copyright Verification on GAN Models	No
	SAC [179]	2022	Wrongly-classified Samples/ CutMix-augmented Samples	Correlation Matrices of Model Predictions	A Threshold on Metric Distances	White-box	Black-box	Copyright Verification on Classification Models	SAC-w:Y SAC-c:N
	DI [144]	2021	Verifier-private Training Samples	Train-Test Margin on Model Predictions	Confidence Regressor & Hypothesis Testing	White&Black	White&Black	Copyright Verification on Classification Models	No
	SSL-DI [180]	2022	Verifier-private Training Samples	Data Density Margin on Representation Domain	GMM Density Estimator & Hypothesis Testing	Black-box	Black-box	Copyright Verification on Self-supervised Models	No
	TeacherFP [181]	2022	Synthetic Samples by min representation distances	Three belief metrics using Hard Predictions	"One-of-the-Best" MR from Supporting Sets	White-box	Black-box	Teacher Identification for Transfer Learning	No
	ZEST [182]	2022	Random samples & Their Constructed Neighbors	Model Weights from LIME Approximation	A Distance Threshold on LIME Weights	White&Black	Black-box	Copyright Verification on Classification Models	No
	FPNet [183]	2022	-	Spatial Spectrum	A Detector Network	None-box	None-box	Generated Image Detection for Unseen Models	No

ratios to construct positive models; and the negative models come from three sources: i) relatively small-scale DNNs on the same training dataset; ii) public pretrained models from online sources finetuned on the training dataset; iii) some irrelevant public models. UAP-FP [31] trains a fingerprint encoder using supervised contrastive loss to represent the fingerprint similarity. Moreover, model preparation is also applied in PIH&TLH [168], the method based on weight comparison, to train the hash generators, which can be categorized as test metric designs.

6.3.2 *Test Case Generation*

As a critical component of DNN fingerprinting, many research works have explored various possibilities to generate conferrable test cases. The word "conferrable" has two meanings: i) The first is robustness. It refers that the protected model and its pirated versions should have similar model behaviors on the generated test cases. These cases need to defend various attack methods like IP detection, evasion and removal described in Section 4. ii) The second is uniqueness, *i.e.*, suitable transferability. It refers that irrelevant models' behaviors should be distinguished from the protected model's behaviors on the generated test cases.

Most methods generate test cases by the adversarial attack or its variants, like the Fast Gradient Sign Method (FGSM) [185], Projected Gradient Descent (PGD) [186], or Carlini & Wagner (C&W) [187]. IPGuard [28], the first study for DNN fingerprint, generates the data points near the decision boundary of the target

model. The optimization problem can be formulated as

$$\min_{\boldsymbol{x}\in\mathcal{X}} \mathrm{ReLU}(g_i(\boldsymbol{x})-g_j(\boldsymbol{x})+k)+\mathrm{ReLU}(\max_{t\neq i,j} g_t(\boldsymbol{x})-g_i(\boldsymbol{x})) \quad (38)$$

where i and j are randomly sampled labels and $\mathrm{ReLU}(x) = max(0, x)$, and g denotes the logits of the target classifier. The hyperparameter k is used to balance between robustness and uniqueness. Following this idea, AFA [170] proposes adversarial marks, *i.e.,* the adversarial samples to mimic the logits and the predicted labels of the target samples randomly chosen from the target class. Wang *et al* [177] leverage the Deepfool [188] method to extract the adversarial samples with minimal perturbation by projecting seed samples onto the decision boundary.

The above methods only consider the decision boundary of the protected model $\mathcal{M}$. However, the decision boundary of $\mathcal{M}$ may partially coincide with irrelevant models but not the piracy models. To this end, CEM [29] constructs conferrable test cases via the conferrable ensemble model $\mathcal{M}_E$ that consists of the positive models $\mathcal{M}_+$ and the negative models $\mathcal{M}_-$, defined as:

$$\begin{aligned}\mathrm{Surr}(\mathcal{M}_+,\boldsymbol{x}) &= \frac{1}{|\mathcal{M}_+|}\sum_{M_+\in\mathcal{M}_+}\mathrm{Dropout}(M_+(\boldsymbol{x})),\\ \mathrm{Ref}(\mathcal{M}_-,\boldsymbol{x}) &= \frac{1}{|\mathcal{M}_-|}\sum_{M_-\in\mathcal{M}_-}\mathrm{Dropout}(M_-(\boldsymbol{x})),\\ \mathcal{M}_E(\boldsymbol{x};\mathcal{M}_{\{+,-\}}) &= \mathrm{softmax}(\mathrm{Surr}(\mathcal{M}_+,\boldsymbol{x})(1-\mathrm{Ref}(\mathcal{M}_-,\boldsymbol{x}))).\end{aligned} \quad (39)$$

It measures the conferrable score of the sample $\boldsymbol{x}$, where Surr and Ref depict the prediction confidence of $\boldsymbol{x}$ on $\mathcal{M}_+$ and $\mathcal{M}_-$ respectively. Given the seed samples $\mathbf{X}_s$ and their perturbed version $\mathbf{X}_\Delta = \mathbf{X}_s + \Delta(\Delta \leq \epsilon)$, a loss function is formulated as:

$$\begin{aligned}\mathcal{L}_(\mathbf{X}_s,\mathbf{X}_\Delta) = \lambda_1 CE(1,\max_t[\mathcal{M}_E(\mathbf{X}_\Delta)_t]) - \lambda_2 CE(\mathcal{M}(\mathbf{X}),\\ \mathcal{M}(\mathbf{X}_\Delta)) + \lambda_3 CE(\mathcal{M}(\mathbf{X}_\Delta),\mathrm{Surr}(\mathcal{M}_+,\mathbf{X}_\Delta)),\end{aligned} \quad (40)$$

where CE denotes the cross-entropy loss. The first item maximizes $\mathcal{M}_E(\mathbf{X}_\Delta)_t$, *i.e.,* the conferrable score for the target class t. The second item maximizes the prediction difference between the seed samples $\mathbf{X}_s$ and their perturbed version $\mathbf{X}_\Delta$. The third item minimizes the prediction difference between the protected model $\mathcal{M}$ and the positive models $\mathcal{M}_+$. From $\mathbf{X}_\Delta$, the samples with high conferrable scores are selected as test cases.

For a similar purpose, Wang *et al* propose Characteristic Examples [171] to generate test cases with high robustness and low transferability in a data-free way. In this work, test cases are initialized by random seeds rather than real samples. For high robustness, this work injects random noises into the model parameters. For low transferability, it uses a high-pass filter to get the high-frequency component of samples at each iteration.

The decision boundary depicted by adversarial examples shows lacking robustness for model modification and adversarial defenses. Therefore, MetaFinger [172] extracts test cases from inner-decision area shared by positive models instead of the boundary. It is realized via a triplet loss $\mathcal{L}$ formulated as

$$\begin{aligned}\mathcal{L}(\mathbf{X}_s) = &\frac{1}{N}\sum_{i=1}^{N} KL(p_a(\mathbf{X}_s),p_i(\mathbf{X}_s))\\ &-\lambda\frac{1}{N}\sum_{i=1}^{N} KL(p_a(\mathbf{X}_s),n_i(\mathbf{X}_s))\end{aligned} \quad (41)$$

where the anchor p_a and the positive points p_i are selected from the positive models, and the negative points n_i are selected from the negative models. The target of $\mathcal{L}$ is to find the test cases $\mathbf{X}_s$ to minimize the KL divergence between $p_a(\mathbf{X}_s)$ and $p_i(\mathbf{X}_s)$ while maximizing the KL divergence between $p_a(\mathbf{X}_s)$ and $n_i(\mathbf{X}_s)$.

All of the above methods apply local adversarial attacks or their variants, *i.e.,* , giving a specific perturbation for each sample. However, local adversarial samples can only capture local geometric information of decision boundaries with high randomness. To this end, UAP-FP [31] constructs a Universal Adversarial Perturbation (UAP) Δ of the protected model that is normal to most of the vectors on the decision boundary. Furthermore, to profile the decision boundary better, UAP-FP uses K-means to cluster the training samples according to the representation vectors; then picks a sample from each cluster into the seed samples $\mathbf{X}_s$. The UAP Δ is constructed on $\mathbf{X}_s$. UAP-FP also proposes Multiviews Fingerprints Augmentation that combines K nearest neighbors for each $\boldsymbol{x} \in \mathbf{X}_s$ into the extended test cases $\mathbf{X}_s^K$. The soft predictions are computed as $\{\boldsymbol{y}_s^K, \boldsymbol{y}_\Delta^K\} = g(\{\mathbf{X}_s^K, \mathbf{X}_\Delta^K\})$ where $\mathbf{X}_\Delta^K = \mathbf{X}_s^K + \Delta$.

Most methods for test case generation rely on concepts of adversarial samples and decision boundary not explicitly existing in other tasks like regression and generative models. For task-agnostic models, TAFA [30] generates test cases based random seeds $\mathbf{X}_s$ instead of training samples, as

$$\max_\Delta \|\Delta\|_1,\ \text{s.t.}\boldsymbol{A}(\mathbf{X}_s+\Delta)=\boldsymbol{A}(\mathbf{X}_s),\ \mathbf{X}_s+\Delta\in\mathcal{X}, \quad (42)$$

where $\boldsymbol{A}(\cdot)$ denotes the activation map of the first ReLU layer. If the first layer of the protected model is a fully-connected layer or a convolutional layer, the constraint of Equ. 42 is linear inequalities about Δ. Thus, TAFA reformulates the problem as a linear programming problem to solve efficiently, as:

$$\begin{aligned}\max_{\boldsymbol{\Delta},\mathbf{Z}} \sum\nolimits_j Z_j,\ \text{s.t.}\ \forall j,\ -Z_j\leq\Delta_j\leq Z_j,\\ \forall\boldsymbol{x}_j\in\mathbf{X}_s,\ (2\boldsymbol{A}(\boldsymbol{x}_j)-1)(\langle\boldsymbol{\Theta},\boldsymbol{x}_j+\Delta_j\rangle+\boldsymbol{b})>\epsilon,\end{aligned} \quad (43)$$

where Z_j is the slack variable of each Δ_j, and $\boldsymbol{\Theta}$ denotes the weights of the first layer. The positive constant ϵ controls the distance from the boundary; thus a larger ϵ pushes the generated cases away from the boundary. From a similar motivation, MetaV [33] jointly optimizes the test cases $\mathbf{X}_s$ and the meta classifier $\mathcal{V}$'s weights as the following optimization problem:

$$\begin{aligned}\max_{\mathbf{X}_s,\mathcal{V}} \log p_+(\mathcal{M}(\mathbf{X}_s)) + \frac{1}{|\mathcal{M}_+|}\sum_{M_+\in\mathcal{M}_+}\log p_+(M_+(\mathbf{X}_s))\\ + \frac{1}{|\mathcal{M}_-|}\sum_{M_-\in\mathcal{M}_-}\log p_-(M_-(\mathbf{X}_s)),\end{aligned} \quad (44)$$

where $\{p_+(\mathcal{M}(\mathbf{X}_s)), p_-(\mathcal{M}(\mathbf{X}_s))\} = \mathcal{V}(\mathcal{M}(\mathbf{X}_s))$, the prediction of the meta-classifier. This problem can be optimized by PGD.

Modeldiff [34] generates pairwise test cases, *i.e.,* the seed samples $\mathbf{X}_s$ and the perturbed version $\mathbf{X}_\Delta$ by maximizing the test scores s on the protected model $\mathcal{M}$ as $\max_{\mathbf{X}_\Delta} s(\mathbf{X}_s, \mathbf{X}_\Delta)$. The test score s is defined as:

$$\begin{aligned}\mathrm{divergence}(\mathbf{X}_s,\mathbf{X}_\Delta) &= \underset{i=1,\ldots,|\mathbf{X}_s|}{\mathrm{mean}}\{\|\mathcal{M}(\boldsymbol{x}_i)-\mathcal{M}(\boldsymbol{x}_i')\|_2\}\\ \mathrm{diversity}(\mathbf{X}_\Delta) &= \underset{\boldsymbol{x}_i',\boldsymbol{x}_j'\in\mathbf{X}_\Delta}{\mathrm{mean}}\{\|\mathcal{M}(\boldsymbol{x}_i')-\mathcal{M}(\boldsymbol{x}_j')\|_2\}\\ s(\mathbf{X}_s,\mathbf{X}_\Delta) &= \mathrm{divergence}(\mathbf{X}_s,\mathbf{X}_\Delta)+\lambda\cdot\mathrm{diversity}(\mathbf{X}_\Delta)\end{aligned} \quad (45)$$

where the first line depicts the intra-input distance, *i.e.,* the distance of soft predictions between $\mathbf{X}_s$ and $\mathbf{X}_\Delta$; the second line depicts the inter-input diversity, *i.e.,* the diversity of the soft predictions of $\mathbf{X}_\Delta$. Since gradients cannot be obtained in black-box registration scenarios, Modeldiff designs an iterative algorithm for black-box generation inspired by mutation testing [189] and black-box adversarial attacks [190]: i) constructing some neighbors of low-score samples by random perturbation; ii) stepping into the one with a higher test score.

DeepJudge [35] propose test case generation methods for both the white-box and black-box verification scenarios. Adversarial attack methods like FGSM, PGD and C&W are directly used for the black-box scenarios. For their proposed text metrics in the white-box scenarios, DeepJudge generates one test case for each neuron covered by the metrics. Given a seed sample and a selected neuron, DeepJudge adjusts the sample to maximize the

neuron's activation by PGD until the activation value surpasses a threshold k pre-computed using training samples.

Aiming at the dynamic model that exhibits varied time costs on input samples, Dong *et al* [173] propose a novel view to fingerprint multi-exit models, *i.e.,* inference time. Hence, the test cases are generated by maximizing the inference time. Given a model with m exits $\{g_i|i=1...m\}$, the loss function is defined as

$$\mathcal{L}(\mathbf{X}_s)=\sum_{i=1}^{m-1} KL(g_i(\mathbf{X}_s),\boldsymbol{u})-KL(g_m(\mathbf{X}_s),\boldsymbol{u}), \quad (46)$$

where $\boldsymbol{u}$ is the uniform vector of the same length as $g_i(\mathbf{X}_s)$. The loss pushes the internal exits' soft predictions towards the uniform vector $\boldsymbol{u}$ while pushing the last exit's prediction away from $\boldsymbol{u}$. Thus, when inputting the cases $\mathbf{X}_s$, the protected model will output from the last exit but irrelevant models will not.

Against transfer learning, TeacherFP [181] proposes to fingerprint teacher models to judge from which teacher model a student model is learned. Given the probing samples $\mathbf{X}_s$ and the feature extractor g_T, the cases $\widetilde{\mathbf{X}}_s$ are generated from random seeds by approximating the probing samples' features $g_T(\mathbf{X}_s)$, as

$$\min_{\widetilde{\mathbf{X}}_s}\|g_T(\widetilde{\mathbf{X}}_s)-g_T(\mathbf{X}_s)\|_2,\ \text{s.t.}\ \widetilde{\mathbf{X}}_s\in\mathcal{X}, \quad (47)$$

where $\mathcal{X}$ denotes the feasible solution space. This constrained optimization can be solved by C&W with a new variable $\mathbf{W}$, as

$$\min_{\mathbf{W}}\|g_T(\tanh(\mathbf{W}))-g_T(\mathbf{X}_s)\|_2,\ s.t.\ \widetilde{\mathbf{X}}_s=\tanh(\mathbf{W}). \quad (48)$$

He *et al* [174] propose the sensitive-sample fingerprinting for Integrity Verification. Here, the test cases $\mathbf{X}_s$ are generated by maximizing the predictions on $\mathbf{X}_s$ between the protected model $\mathcal{M}$ and its modified version $\mathcal{M}_\Delta$ as $\max_{\mathbf{X}_s}\|\mathcal{M}(\mathbf{X}_s)-\mathcal{M}_\Delta(\mathbf{X}_s)\|$, which is equivalent to minimizing the model gradients as

$$\min_{\mathbf{X}_s}\left\|\frac{\partial\mathcal{M}(\mathbf{X}_s;\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\right\|_F^2,\ \text{s.t.}\ \|\mathbf{X}_s-\mathbf{X}_0\|<\epsilon,\ \mathbf{X}_s\in\mathcal{X}, \quad (49)$$

where $\|\cdot\|_F^2$ denotes the Frobenius norm of a matrix, and the constraint $\|\mathbf{X}_s-\mathbf{X}_0\|<\epsilon$ is to make $\mathbf{X}_s$ look like the natural samples $\mathbf{X}_0$ to defend against IP evasion attacks. Then, a Maximum Active Neuron Cover (MANC) algorithm is designed to select some test cases maximizing the number of activated neurons from $\mathbf{X}_s$. Furthermore, for Public Integrity Verification in double black-box scenarios, PublicCheck [175] designs encysted samples generated by attribute manipulation along a semantic feature axis. First, a disentangled auto-encoder model $\mathcal{G}$ is trained on a subset of training samples of the protected model for attribute-level and abstraction-level disentanglement. Then, it uses the model $\mathcal{G}$ to manipulate the seed samples $\mathbf{X}_s$ in the latent space until causing prediction changes, records the latent vectors $\mathbf{Z}_\Delta$, and randomly samples the augmented latent vectors $\mathbf{Z}'_\Delta$ around $\mathbf{Z}_\Delta$. The augmented candidates $\mathbf{X}_\Delta$ are reconstructed using the decoder of $\mathcal{G}$ as $\mathbf{X}_\Delta=\mathcal{G}_{DE}(\mathbf{Z}'_\Delta)$. Against IP Evasion attacks, the encysted samples are selected from $\mathbf{X}_\Delta$ via a smoothness metric to ensure pixel-smooth added patterns $\mathbf{X}_\Delta-\mathbf{X}_s$. Wang *et al* [176] propose Functionality Verification, *i.e.,* a variety of Integrity Verification that detects adversarial third-party attacks like transfer learning and backdoor attacks while enabling model compression. Here, the test cases $\mathbf{X}_s$ are randomly initialized and updated by a min-max optimization as:

$$\min_{\mathbf{X}_s}\max_{\Delta_m}\mathcal{L}(\mathbf{X}_s,\mathcal{M}+\Delta_m,\boldsymbol{y}_t),\ \text{s.t.}\ \mathbf{X}_s\in\mathcal{X},\ \Delta_m<\epsilon, \quad (50)$$

where the inner max-optimization constructs the perturbed version $\mathcal{M}+\Delta_m$ of the target model, and the outer min-optimization obtains the adversarial samples $\mathbf{X}_s$ of $\mathcal{M}+\Delta_m$ to the target label $\boldsymbol{y}_t$. The two processes are performed alternatively.

Test cases can be selected from a given dataset with simple or even no additional processing. SAC [179] proposes two methods to select test cases: i) SAC-w selects the samples wrongly classified by the protected or pirated models but correctly classified by irrelevant models, and ii) SAC-c applies CutMix, a data augmentation method, on randomly selected samples. MeFA [32] uses a meta-classifier $\mathcal{V}$ to select some in-distribution data samples as test cases. Here, the meta-classifier $\mathcal{V}$ is trained on the soft predictions of prepared models. If the result of $\mathcal{V}(\mathcal{M}_{ref+}(\boldsymbol{x}))$ is "1" (in-distribution), the sample $\boldsymbol{x}$ will be selected.

6.3.3 Test Metric Design

Most of the current works [28], [29], [31]–[33], [170]–[172], [175], [177] directly use model outputs like soft/hard predictions $g(\mathbf{X}_s)$ (or $g(\{\mathbf{X}_s,\mathbf{X}_\Delta\})$) on test cases $\mathbf{X}_s$ (or pairwise cases $\{\mathbf{X}_s,\mathbf{X}_\Delta\}$) as the test metrics. Soft predictions have more information about model behavior and can support more ways for fingerprint comparison, but sometimes may be unavailable in the black-box scenarios. If necessary, label smoothing can be applied to transform hard predictions to a soft probability vector.

In general, if the generated test cases are good enough, such test metrics have the discriminability to judge piracy/irrelevant models. It requires the test cases are not too close to the decision boundary or the class centroid. The former leads to a high false negative rate and the vulnerability against IP removal attacks, while the latter may misjudge irrelevant models as piracy models. However, it is hard to satisfy. In addition, for some tasks like self-supervised learning or transfer learning, fingerprints should be registered in a representation domain rather than model predictions. Thus, some works develop more possibilities for test metric design to realize a better discriminability, generally through processing the model predictions or neuron activation on test cases. Here, we describe these schemes in details.

To our knowledge, Modeldiff [34] is the first design for test metrics, named Decision Distance Vectors (DDV). A DDV captures the decision pattern of a model on pairwise test cases $\{\mathbf{X}_s,\mathbf{X}_\Delta\}$. $\mathbf{X}_s$ and $\mathbf{X}_\Delta$ denote the seed samples and their perturbed version, respectively. The DDV of the model g is defined as

$$DDV_g=\{dist(g(\boldsymbol{x}),g(\boldsymbol{x}'))|\forall\boldsymbol{x}\in\mathbf{X}_s\}, \quad (51)$$

where $\boldsymbol{x}'\in\mathbf{X}_\Delta$ is the perturbed version of $\boldsymbol{x}$, and $dist$ is the cosine distance if $g(\boldsymbol{x})$ is a 1-D array.

TeacherFP [181] proposes three belief metrics to select the model that the suspected model $\hat{g}$ learns from, given m candidate teacher models $\mathcal{G}=\{g_i|i=1...m\}$: Matching Rate (MR), Eccentricity (Ecc), and Empirical Entropy (EE), defined as

$$\begin{aligned}&\text{MR}_g=|\mathbf{X}_m|/|\mathbf{X}_s|,\ \mathbf{X}_m=\{\boldsymbol{x}|g(\boldsymbol{x})=\hat{g}(\boldsymbol{x}),\forall\boldsymbol{x}\in\mathbf{X}_s\},\\&\text{Ecc}=(\max_1(\boldsymbol{v})-\max_2(\boldsymbol{v}))/\sigma(\boldsymbol{v}),\ \boldsymbol{v}=\{\text{MR}_g|\forall g\in\mathcal{G}\},\\&\text{EE}=-\sum\nolimits_{y\in\mathcal{Y}}\hat{p}_y\log\hat{p}_y,\ \hat{p}_y=\text{Prob}(g^*(\mathbf{X}_m)=y),\end{aligned} \quad (52)$$

where $g^*=\arg\max_g\text{MR}_g$ denotes the model with the maximal MR in candidate teacher models $\mathcal{G}$, $\max_k$ is to find the k-th largest value in the set, and $\sigma(\boldsymbol{v})$ refers to the standard deviation of $\boldsymbol{v}$. MR reflects the similarity of hard predictions on the test cases $\mathbf{X}_s$; Ecc measures the prominence degree of the extrema in a set; and EE depicts how much information is given to make the decision.

DeepJudge [35] designs more comprehensive and fine-grained test metrics on three levels: i) *Property-level* including Robustness Distance (RobD); ii) *Neuron-level* including Neuron Output Distance (NOD) and Neuron Activation Distance (NAD); and iii) Layer-level including Layer Activation Distance (LAD) Jensen-Shanon Distance (JSD). The property-level metrics are a set of model properties used to characterize model similarity, like RobD that depicts models' difference in adversarial robustness:

$$RobD(g,\hat{g},\mathbf{X}_s)=\sum_{\boldsymbol{x}\in\mathbf{X}_\Delta}|(\mathbb{1}(g(\boldsymbol{x})\neq y_s)-\mathbb{1}(\hat{g}(\boldsymbol{x})\neq y_s)|, \quad (53)$$

where $\mathbf{X}_\Delta$ denotes the adversarial samples derived from the seed samples $\mathbf{X}_s$, y_s denotes the true labels, and $\mathbb{1}$ denotes the indicator function. The output of each neuron in a model follows its own statistical distribution, which could be utilized as test metrics. Given the i-neuron output function of the protected model and the suspected model on the l-layer $\phi_{l,i}, \hat{\phi}_{l,i}$, the neuron-level metrics, NOD and NAD, are defined as

$$\begin{aligned} NOD(\phi_{l,i}, \hat{\phi}_{l,i}, \mathbf{X}_s) &= \sum\nolimits_{\boldsymbol{x}\in\mathbf{X}_\Delta} |\phi_{l,i}(\boldsymbol{x}) - \hat{\phi}_{l,i}(\boldsymbol{x})|, \\ NAD(\phi_{l,i}, \hat{\phi}_{l,i}, \mathbf{X}_s) &= \sum\nolimits_{\boldsymbol{x}\in\mathbf{X}_\Delta} |S(\phi_{l,i}(\boldsymbol{x})) - S(\hat{\phi}_{l,i}(\boldsymbol{x}))|, \end{aligned} \tag{54}$$

where the step function $S(\phi_{l,i}(\boldsymbol{x}))$ returns 1 if $\phi_{l,i}(\boldsymbol{x})$ is greater than a threshold, 0 otherwise. The layer-level metrics, including LOD, LAD and JSD, depict the model difference in layer outputs:

$$\begin{aligned} LOD(f^l, \hat{f}^l, \mathbf{X}_s) &= \sum\nolimits_{\boldsymbol{x}\in\mathbf{X}_\Delta} \|f^l(\boldsymbol{x}) - \hat{f}^l(\boldsymbol{x})\|_2, \\ LAD(f^l, \hat{f}^l, \mathbf{X}_s) &= \sum\nolimits_{i=1}^{|N_l|} NAD(\phi_{l,i}, \hat{\phi}_{l,i}, \mathbf{X}_s), \\ JSD(f^L, \hat{f}^L, \mathbf{X}_s) &= \sum\nolimits_{\boldsymbol{x}\in\mathbf{X}_\Delta} \{KL(f^L(\boldsymbol{x}), \boldsymbol{q}) + KL(\hat{f}^L(\boldsymbol{x}), \boldsymbol{q})\} \end{aligned} \tag{55}$$

where $f^l/\hat{f}^l$ denotes the l-layer output function of the protected/suspected model, $\boldsymbol{q} = (f^L(\boldsymbol{x}) + \hat{f}^L(\boldsymbol{x}))/2$, and L is the index of the output layer. JSD quantifies the model similarity in output distributions; thus it is robust against model extraction attacks. Note that RobD and JSD are designed for the black-box verification scenarios while the others, NOD, NAD, LOD and LAD, are designed for the white-box scenarios.

Dataset Inference (DI) [144] leverages the train-test margin on model predictions as a test metric. The intuition is that, the protected model exhibits more confidence in its training samples rather than in unseen testing samples. It is depicted by performing gradient descent optimization of the following objective:

$$\arg\min_\Delta \delta_{\boldsymbol{t}}(\mathbf{X}_s, \mathbf{X}_s + \Delta), \text{ s.t. } g(\mathbf{X}_s + \Delta) = \boldsymbol{t}, \forall \boldsymbol{t} \neq \boldsymbol{y}, \tag{56}$$

where $\boldsymbol{y}$ is the hard labels of the selected verifier-private training samples $\mathbf{X}_s$, and δ is the samples' distance metric. Then, the prediction margin, *i.e.,* the minimal distance to the decision boundary with T-classes $\{\delta_{\boldsymbol{t}}|t = 1...T\}$, is obtained. For the black-box registration scenarios, the gradients cannot be obtained; thus, DI gives a fixed perturbation Δ_0 and minimizes k as

$$\arg\min_k \delta_{\boldsymbol{t}}(\mathbf{X}_s, \mathbf{X}_s + k\Delta_0), \text{ s.t. } g(\mathbf{X}_s + k\Delta_0) = \boldsymbol{t}, \forall \boldsymbol{t} \neq \boldsymbol{y}. \tag{57}$$

The theoretical proof shows that DI succeeds nearly every time since DI aggregates the signal over multiple samples. Following this idea, Adam *et al* propose an extended DI for Self-Supervised Learning models (SSL-DI) [180]. DI is an attractive way to protect SSL models because it is costly to retrain large SSL models [191] and the performance will affect all downstream tasks. However, the raw DI method [144] relies on computing the distances between the samples and the decision boundary: i) SSL encoders trained on unlabeled data have no explicit decision boundaries; ii) SSL models do not have enough overfitting to show a clear train-test margin on predictions. Thus, SSL-DI uses the data density on the representation domain as the margin. Its key principle is that the log-likelihood of output representation on training samples is larger than on testing samples for the positive models, however, this is not the case for the negative models. The log-likelihood is depicted by density estimation using Gaussian Mixture Models.

Most existing test metrics are designed for specific tasks like classification, highly relying on task-specific model predictions. Thus, for task-agnostic models independent from specific tasks and training samples, TAFA [30] design a method derived from the activation of the first ReLU layer $\boldsymbol{A}$. It depicts the distance from the random seeds $\mathbf{X}_s$ to the boundary of linear regions (*i.e.,* the adversarial version $\mathbf{X}_\Delta$ of $\mathbf{X}_s$). Based on the first-ReLU activation on a suspected model, TAFA computes the normalized Hamming distance as the confidence score, as

$$\text{MR}_{\text{wb}} = 1 - \frac{1}{d_1|\mathbf{X}_s|} \sum_{\boldsymbol{x}\in\mathbf{X}_s} tr(abs(\boldsymbol{A}(\boldsymbol{x}) - \boldsymbol{A}(\boldsymbol{x}_\Delta))), \tag{58}$$

where d_1 is the dimension of a sample's ReLU activation and $\boldsymbol{x}_\Delta$ is the adversarial version of $\boldsymbol{x}$ derived from Equ. 43. For black-box verification, *Pointsonline* is used to approximate this metric.

The performance on robustness and uniqueness of most test metrics depends heavily on excellent test cases, while the case-free methods based on weight comparison are hard to support cross-architecture verification and black-box scenarios. To this end, ZEST [182] introduces the Local Interpretable Model-Agnostic Explanations (LIME) algorithm as test metrics. First, ZEST selects some training samples $\mathbf{X}_s$, and for each sample $\boldsymbol{x}_i \in \mathbf{X}_s$, a set of its nearby samples $\mathbf{X}_i^\Delta$ is constructed using the Quickshift method [192]. Then, for each sample set $\mathbf{X}_i^\Delta$, Zing trains a local linear regression model $h_i : \mathbf{X}_i^\Delta \rightarrow g(\mathbf{X}_i^\Delta)$ to approximate the model's local behavior on $\mathbf{X}_i^\Delta$. Finally, the weights of all the regression models are concatenated as an IP signature approximating the global model behavior.

The existing fingerprinting schemes based on transferable adversarial examples are sensitive to transfer learning scenarios and defense strategies against adversarial attacks. Thus, SAC [179] designs the pairwise sample correlation as the test metric. Given the test cases $\mathbf{X}_s = \{\boldsymbol{x}_i|i = 1...|\mathbf{X}_s|\}$ and their model predictions $\mathcal{O} = \{\boldsymbol{o}_i|\boldsymbol{o}_i = g(\boldsymbol{x}_i), i = 1...|\mathbf{X}_s|\}$, SAC computes a model-specific correlation matrix $\mathbf{C} \in \mathbb{R}^{|\mathbf{X}_s|\cdot|\mathbf{X}_s|}$ as

$$\mathbf{C} = \{c_{i,j}|c_{i,j} = corr(\boldsymbol{o}_i, \boldsymbol{o}_j), i, j = 1...|\mathbf{X}_s|\}, \tag{59}$$

where $corr$ could be cosine similarity or Gaussian RBF, as

$$\begin{aligned} corr(\boldsymbol{o}_i, \boldsymbol{o}_j) &= cos(\boldsymbol{o}_i, \boldsymbol{o}_j) = \frac{\boldsymbol{o}_i^T \boldsymbol{o}_j}{\|\boldsymbol{o}_i\|\|\boldsymbol{o}_j\|} \text{ or} \\ corr(\boldsymbol{o}_i, \boldsymbol{o}_j) &= RBF(\boldsymbol{o}_i, \boldsymbol{o}_j) = \exp(-\frac{\|\boldsymbol{o}_i - \boldsymbol{o}_j\|_2^2}{2\delta^2}). \end{aligned} \tag{60}$$

Note that $\boldsymbol{o}_i$ is a probability vector. For those scenarios that can only obtain the suspected model's hard label y, label smoothing is applied to transform the hard label into a probability vector, as $\boldsymbol{o} = (1-\epsilon)\eta(y) + \epsilon\boldsymbol{u}$, where $\boldsymbol{u}$ is a uniform vector and $\eta(y)$ is a one-hot encoding for the label y.

Inference time could be also used as test metrics [173]. In the future, more possibilities for test metric designs, especially with interpretability or theoretical constraints, need to be explored to cover various emerging tasks and multi-view demands.

6.3.4 Fingerprint Comparison

Test metric thresholds are commonly used for fingerprint comparison [28]–[30], [170]–[173], [175], [177], [179], [181], [182]. Most methods based on hard predictions use a threshold on the Matching Rate (MR in Equ. 52) to judge piracy models. An MR measures the proportion of the matching set, *i.e.,* the test cases that have the same predicted labels between the protected model g and the suspected model $\hat{g}$. If the MR between g and $\hat{g}$ is larger than a threshold ϵ, the suspected model $\hat{g}$ can be regard as piracy. On this basis, TeacherFP [181] develops a "one-of-the-best" strategy for transfer learning.It first selects the supporting set by removing the test cases with the maximum occurrence from the matching set. Then, given a suspected model $\hat{g}$ and some candidate teacher models $\mathcal{G}$, TeacherFP selects the victim model with the maximum MR on the supporting set from $\mathcal{G}$.

Simple threshold comparison is also used for elaborate test metrics. For example, Modeldiff uses a threshold on the cosine similarity of DDVs (Equ. 51); TAFA uses a threshold on the distances from test cases to region boundaries as Equ. 58;

Dong *et al* [173] use a threshold on EEC AUC scores; DeepFool-FP [177] uses a confidence threshold on a probability derived from Bayesian Theorem; SAC [179] uses a distance threshold on sample correction matrices as Equ.59; ZEST [182] use a distance threshold on LIME weights. DeepJudge [35] determines a specific threshold for each test metric and designs a majority voting strategy: the model is judged pirated if more than half of the metrics of the suspected model exceed the given thresholds. It is worth mentioning that, though a simple voting method can work in DeepJudge, more exploration about multi-metric decision is valuable in practical scenarios.

Some advanced methods apply machine learning models for fingerprint comparison. SSL-DI [180] trains Gaussian Mixture Models (GMM) on data representations to estimate the data density in the latent space. MeFA [33], MetaV [33], and DI [144] train binary fingerprint classifiers $\mathcal{V}$ whose outputs are close to "1" given piracy fingerprints, and "0" given irrelevant fingerprints. Therein, MeFA and MetaV use soft predictions on test cases as inputs, while DI uses the prediction margin as Equ. 56. Similarly, UAP-FP [31] uses augmented fingerprints to train an encoder to represent fingerprints into a latent space and compares the fingerprints by the cosine similarity of their representations. To leverage the label information, UAP-FP trains the two-branch structure of SimCLR [193] via a supervised contrastive loss as

$$\mathcal{L} = \sum_{i \in I} -\frac{1}{|\Psi(i)|} \sum_{\mu \in \Psi(i)} \log \frac{\exp(\cos(\boldsymbol{z}_i, \boldsymbol{z}_\mu)/\tau)}{\sum_{\nu \in \{I \setminus i\}} \exp(\cos(\boldsymbol{z}_i, \boldsymbol{z}_\nu)/\tau)} \quad (61)$$

where $\boldsymbol{z} = E(\{g(\mathbf{X}_s), g(\mathbf{X}_\Delta)\})$ is the output of the encoder E; I denotes the batch index; $\Psi(i) = \{\mu | \mu \in \{I \setminus i\}, \tilde{y}_\mu = \tilde{y}_i\}$ denotes the fingerprint indexes with the same label of the i-th fingerprint.

The performance of fingerprint comparison is sensitive to the choice of threshold, even for fingerprint classifiers that output soft probabilities. The threshold can be preset as a empirical value [28], [29], [171], [172], or be determined in a data-driven way. For example, AFA [170] constructs some positive models (ghost networks) by Dropout, and uses half of the average MR of the ghost networks as the threshold; SAC [179] uses the average of the correlation scores of negative and positive models; and MeFA [33] use by the average of the fingerprint classifier's soft predictions on test cases. Note that the MR threshold is set a 1 for the integrity verification task [174], [175].

In addition, Hypothesis Testing is often used to avoid manual threshold setting. DI [144] forms a two-sample t-Test on the distribution on confidence score vectors. First, it extracts the prediction margins $\boldsymbol{\delta}$ and $\boldsymbol{\delta}'$ of some training samples on the protected and suspected model as Equ. 56, respectively; then inputs $\boldsymbol{\delta}, \boldsymbol{\delta}'$ to the confidence regressor (a fingerprint classifier) to get confidence score vectors $\boldsymbol{c}$ and $\boldsymbol{c}'$. Next, a two sample T-test is formed on $\boldsymbol{c}$ and $\boldsymbol{c}'$. If the p-value for the one-sided hypothesis $H_0 : \bar{\boldsymbol{c}}' < \bar{\boldsymbol{c}}$ is smaller than 0.05, the suspected model is judged pirated. Note that $\bar{\boldsymbol{c}}$ is the mean of $\boldsymbol{c}$. UAP-FP uses a similar two-sample t-Test [31] on Ω' and Ω_-, *i.e.*, the fingerprint similarities from suspected models and irrelevant models. The suspected model is judged pirated if Ω' is significantly greater than Ω_-. SSL-DI [180] forms a hypothesis test on the density from GMMs: if the representation density of training samples is significantly greater than that of unseen samples, the test model is stolen. Based on hypothesis tests, DeepJudge [35] designs an ε-difference strategy to select a threshold for each metric λ. It uses one-tailed t-test to get the lower bound LB_λ based on the statistics of negative models at the 99% confidence level and give the threshold as $\alpha_\lambda LB_\lambda$, where α_λ is a relaxing coefficient.

6.3.5 Target Scenarios, Models, & Functions

i) Registration scenarios: Most works require white-box access to register fingerprints because they need model gradients to obtain test cases/metrics. Five works can be used for black-box registration: Modeldiff [34], DI [144], and PublicCheck [175] are inspired by black-box adversarial attacks. They set a fixed perturbation direction and gradually increase the perturbation step length until changing predicted labels. DI/SSL-DI [144], [180] and ZEST [182] directly use private training samples as test cases and require only model outputs to compute test metrics. These methods have been described in detail above and thus we do not repeat them here. In addition, None-box registration is valid for generative models like FPNet [183] without specific test cases.

ii) Verification scenarios: Weight-based methods can be used for only white-box verification while almost all the behavior-based methods work on black-box verification, except the work [173] using white-box inference time. Note that TAFA [30], DeepJudge [35], and DI [144] can extract better test metrics with white-box gradients or hidden layer outputs.

iii) Target functions: Most works focus on copyright verification while some works extend DNN fingerprinting to other tasks: [174] and [175] focus on integrity verification on classification models; [176] proposes robust fingerprints for functionality verification; [168] proposes tampering localization to identify the tampered position of the target model; TeacherFP [181] proposes teacher model fingerprinting attacks against transfer learning to judge which model is a given student model learned from; FPNet [183] proposes generated image detection to judge whether an image is generated or real. Most works focus on the robustness against IP Removal attacks and the distinguishing ability against pirated and irrelevant models. However, few works focus on potential attacks during IP verification, like IP evasion and ambiguity. Some methods can defend against IP detection attacks in part, like DI/SSL-DI [144], [180] using normal training samples, or [175], [174], and [178] extracting test cases close to normal samples. Moreover, interpretable fingerprinting schemes with theoretical guarantees, or fine-grained cause localization of model similarity, have not been focused on.

iv) Target models: Most works are designed for classification models. Besides, other learning tasks are explored: TAFA [30] and MetaV [33] fingerprint task-agnostic models by the first ReLU activation and a fingerprint classifier, respectively; SSL-DI [180] utilizes the train-test margin on representation density to fingerprint encoder models trained from self-supervised learning; Dong *et al* [173] fingerprints multi-exit models via inference time. GAN-FP [178] and FPNet [183] fingerprint generated models. GAN-FP [178] cascades the protected Generative Adversarial Network (GAN) model with a classifier head $\mathcal{V}$ that predicts the class labels of generated outputs. It constructs test latent codes with covert generated outputs against IP Evasion attacks using three ways: i) creating adversarial latent codes whose generated outputs fool the classifier $\mathcal{V}$; ii) using a CE loss to finetune $\mathcal{V}$ to embed invisible backdoors (normal generated outputs with modified labels); iii) using a triplet loss and a fine-grained categorization approach to embed backdoors. The latter two are similar to DNN watermarking, however, they only finetune $\mathcal{V}$ rather than the GAN model. FPNet [183] designs a generated image detection method for unseen generative models. It treats a generative model as a combination of filters and uses the specific spatial spectrum from the upsampling process of image generation as model fingerprints. A fingerprint generator is designed to simulate the fingerprints of unseen generative models, and a fingerprint detection network is trained to determine

whether a given image is generated or real. Vonderfecht *et al* [194] extend fingerprints for generative models to single-image super-resolution (SISR) models and confirm that detectable fingerprints do exist in SISR models, especially using high upscaling factors or adversarial loss.

7 Testing Protocols & Performance

7.1 Existing Testing Protocols

At present, there is no testing protocol for Deep IP defence and attack, which can cover watermarking and fingerprint schemes, span the whole pipeline of IP registration and verification, and test multiple metrics, like fidelity, robustness, and security, in a fair and comprehensive way. Some works try to provide a fair performance comparison for contemporary model watermarking methods. For example, Chen *et al* [41] propose an experimental performance comparison on fidelity, robustness and integrity. Six classic schemes are considered, including inner-component-based [19], [21] and trigger-based watermarking [21], [24]–[26]. The robustness is tested against model fine-tuning attacks, parameter pruning attacks and watermark overwriting attack. [45][4] evaluates the robustness and fidelity of some trigger-based methods. It considers three types of attacks: detection, removal and ambiguity. For IP removal attacks, it proposes adaptive attacks: it consumes that the adversary knows the algorithm but not specific trigger samples, and trains a generative model to mimic trigger samples to repair the predictions. However, these testing protocols are based on certain fixed configurations of watermarking schemes, like the learning rate and the similarity threshold, which is hard to provide a fair comparison. Lukas *et al* [38][5] propose WRT, a watermarking-robust-toolkit, to test DNN watermarking on fidelity, robustness and efficiency. It formulates the embedding and removal of watermarks as a security game. It first constructs a payoff matrix for multiple attack and defence configurations, where an element represents the attack payoff that is zero for unsuccessful attacks and is equal to the accuracy of the surrogate model otherwise. The attacker and defender choose their configurations based on the payoff matrix to achieve Nash Equilibrium. The attacker will win if the normalized watermark accuracy is lower than 0.5 within 5% loss on the task accuracy. [38] mainly tests the robustness against IP removal attacks and implements various attacking means including three types: input processing, model modification and model extraction. More advanced attack and defence strategies, more performance metrics and more complete test pipelines need to be studied continuously with the development of Deep IP, towards commercial testing protocols.

7.2 Performance of representative protection methods

This survey has summarized some performance metrics of Deep IP Protection in Section 2.3. Here, we compare the performance of the current open-source DeepIP solutions on these metrics. The results shown in Table 5 are benchmarked against the experimental results in existing papers that focus on performance comparisons [38], [45]. Moreover, we refer to the theoretical analysis and experimental results in the papers of the corresponding schemes. We define several performance levels which are 'A', 'B',and 'C' from high to low. Specially, '?' means that the metric of the method is without any experimental test and theoretical analysis. 'N/A' stands for unclear, meaning that existing results do not provide a performance level clear enough. The superscript 'T' indicates that the metric is theoretically guaranteed or partially guaranteed.

TABLE 5
The Performance of Deep IP Protection

	Methods	Fidelity			Robustness			Capacity	Security			Others		Code
		Acc	*Eff*	*Risk*	*Pre*	*Mod*	*Ext*	*Upp/Pay*	*Amb*	*Eva*	*Rmv*	*FPR*	*Sca*	**Links**
Watermarking	Uchida *et al* [19]	B	A	-	-	B	C	?/Multi	C	C	B-	A	B	[195]
	Deepmarks [20]	B	A	-	-	B	C	?/Multi	C	C	B-	A	B	[196]
	Deepsigns [21]	B	A	-	-	B+	C	?/Multi	B	B	B	A	B	[196]
	Liu *et al* [96]	A	A	-	-	B+	C	?/Multi	B+	B	B	A	B	[197]
	RIGA [63]	B	A	-	-	B+	C	?/Multi	B-	B+	B	A	B	[198]
	MOVE [102]	A	A	-	-	B+	B	?/One	B+	B+	B+	A	B	[199]
	HufuNet [109]	B	?	-	-	B+	C	?/One	B+	B+	B	A	B	[200]
	Adi *et al* [25]	B+	A	B	B+	B	C	?/One	C	B-	B-	A	B	[196]
	Content [26]	B+	A	C	B+	B	B-	?/One	C	B-	B	A	B	[196]
	Noise [26]	B+	A	C	B+	B+	B-	?/One	C	B-	B	A	B	[196]
	Unralated [26]	B+	A	B-	B+	B	C	?/One	C	B-	B-	A	B	[196]
	Blackmarks [27]	B	A	C	B+	B	B-	?/Multi		N/A		A	B	[196]
	DAWN [150]	B-	A	B-	B+	B-	B+	?/One	C	C	B-	A	B	[196]
	Namba *et al* [66]	B	A	B-	B+	B	C	?/One	?	B+	B	A	B	[201]
	Certified WM [122]	B	A	B-	B+	B+	B-	?/One		N/A		A	B	[202]
	UBW-P [117]	B+	A	A	B+	B	B-	?/One		N/A		A	B	[203]
	Zheng *et al* [115]	B+	A	B-	B+	B+	B-	?/One	B-	B+	B	A	B	[204]
	Jia *et al* [114]	B	A	B-	B+	B+	B+	?/One	B-	B-	B	A	B	[205]
	Merrer *et al* [24]	A	A	A	B+	B+	C	?/One	B-	B-	B	B	B	[206]
	NTL [52]	B+	A	B-	B+	B+	B+	?/One		N/A		A	B	[207]
	UBW-C [117]	B	-	A		N/A		?/One		N/A		A	B	[203]
	CosWM [95]	B	A	?	B+	B+	B+	?/One		N/A		A	B	[208]
	Passport [23]	B-	B+	-	-	B+	B-	?/Multi	B+	B	B+	A	B	[209]
	Passport Norm [107]	B	B+	-	-	B+	B-	?/Multi	B+	B	B+	A	B	[210]
	IPR-GAN [136]	B-	B+	-	-	B+	B-	?/Multi	B+	B	B+	A	B	[211]
	IPR-RNN [108]	B	B+	-	-	B+	B-	?/Multi	B+	B	B+	A	B	[212]
	Lottery [105]	B+	B+	B	-	B	-	?/Multi		N/A		A	B	[213]
	WAFFLE [132]		N/A			N/A		?/One		N/A		N/A		[214]
	FedIPR [133]		N/A			N/A		T/Multi		N/A		N/A		[215]
	SSLGuard [137]		N/A			N/A		?/One		N/A		A	A	[216]
	CoProtector [139]		N/A			N/A		?/One		N/A		N/A		[217]
Fingerprinting	PoL [166]		A^T			N/A		?/One		N/A		A	-	[218]
	IP Guard [28]		A^T			N/A		?/One		N/A		N/A		[219]
	CEM [29]		A^T			N/A		?/One		N/A		N/A		[220]
	MeFA [32]		A^T			N/A		?/One		N/A		N/A		[221]
	MetaFinger [172]		A^T			N/A		?/One		N/A		N/A		[222]
	Modeldiff [34]		A^T			N/A		?/One		N/A		N/A		[223]
	SAC [179]		A^T			N/A		?/One		N/A		N/A		[224]
	DeepJudge [35]		A^T			N/A		?/One		N/A		N/A		[219]
	ZEST [182]		A^T			N/A		?/One		N/A		N/A		[225]
	TeacherFP [181]		A^T			N/A		?/One		N/A		N/A		[226]
	DI [144]		A^T			N/A		?/One		N/A		N/A		[227]
	SSL-DI [180]		A^T			N/A		?/One		N/A		N/A		[228]

Acc: accuracy, Eff: ffficiency, Risk: extra risks; Pre: input preprocessing, Mod: model modification, Ext: model extraction, Upp: theoretical upper mound; Pay: the bit number of payloads; Amb: IP ambiguity, Eva: IP detection & evasion, Rmv: adaptive IP removal; FPR: false positive rate, Sca: the scalability to various models and tasks.

According to Table 5, we highlight the following conclusions:

- The existing benchmark tests have proved that there are no Deep IP solution that can defend against various attack strategies like IP evasion, remove, and ambiguity attacks. It still have a long way towards the guaranteed robustness and security of Deep IP solutions.
- Until now, the vast majority of Deep IP solutions have not been convincingly evaluated to prove their security performance. Especially, there is no testing protocol for the performance of fingerprinting schemes. We hereby advocate more testing protocol and benchmark toolkits to evalutate the performance of Deep IP solutions.
- Only FedIPR [133] has tried to provide a theoretical analysis about watermark capacities and detection rate. It still remains open to exploring capacity bounds of IP signatures.
- Fidelity and robustness are generally traded off against each other. Fingerprinting methods have the optimized fidelity since it requires no model modification, but the robustness is not yet guaranteed. Watermarking methods with higher robustness

4. https://github.com/WSP-LAB/wm-eval-zoo
5. https://crysp.uwaterloo.ca/research/mlsec/wrt, https://github.com/dnn-security/Watermark-Robustness-Toolbox

may have lower fidelity. No empirical or theoretical studies have yet explored the quantitative relationship between fidelity and robustness.
- The task-level scalability of inner-component-based solutions is generally higher than that of input/output-based solutions, because the former usually does not depend on the specific model outputs. However, the former has few scalability across model structures; the internal model components are quite different after cross-structures model extraction.

8 Conclusion and Outlook

This research, which serves as a comprehensive yet contemporary review on deep intellectual property (IP), first discussed the motivations, functions, and requirements, and presented a multi-view taxonomy of recent developments according to the registration procedure, verification scenarios, and IP functions. Then, we summarized the main challenges and threats of deep IP protection, highlighted the recent representative protection methods, and analyzed their merits and demerits. Finally, the evaluation protocols and performance against potential threats were proposed. Nowadays the exponentially increasing training cost and commercial value puts forward urgent and strict requirements for deep IP protection. Although significant progress has been made, as discussed in Section 1, deep IP protection in the open world is still in its infancy and many open challenges remain. Therefore, we identify a number of promising directions for future research.

(1) The Theoretical Analysis and Guarantee of Deep Watermarking or Fingerprinting. At present, only a few schemes [109], [122], [133], [144] propose limited theoretical analysis or certain theoretical guarantees for the performance metrics in specific scenarios, such as the watermark robustness against fine-tuning attacks, or the capacity and the detection rate of parameter-based watermarks. However, this is still far from a complete theory for Deep IP. The fundamental challenge lies in the challenging problem of the interpretability of deep learning. Building such a theory can not only guide the design of Deep IP schemes to guarantee or explain their security but also promote the development of DNN interpretability. Future research directions include but are not limited to: i) how to quantitatively analyze the relationship between watermark capacity, robustness, and model fidelity; ii) how to design Deep IP schemes with theoretical guarantees against various attacks; iii) how to interpret and fine-grained characterize local or global model decision boundaries, and iv) how to determine what causes two models to be similar.

(2) Credible Notarization Mechanism and Platforms of Deep IPs. Most existing Deep IP schemes assume that the IP signatures are verified by the model owner. However, it will lead to the copyright conflict problem, that is, any party can construct its own IP signature using its own algorithm for the same target model. In such cases, it is extremely difficult to determine which side the model belongs to. Thus, a credible verification platform is needed to build a database of secret keys and IP signatures to verify the ownership of models. This platform can be decentralized or owned by a trusted third party, such as a judicial authority. For the coexistence of multiple IP signatures, a mechanism of determining the construction sequence of multiple IP signatures can be designed by leveraging the catastrophic forgetting property of DNN models, and the party that constructs the earliest IP signature of the target model is regarded as the model owner.

(3) Enhancing the Basic Metrics of Deep IP Protection. At present, the area of Deep IP is in the early stage of development and far from mature. None of the solutions has been tested for a long time in a real application environment. Therefore, it is necessary to further strengthen the performance on discriminability, fidelity, and potential new metrics discovered in practical applications. Feasible research directions include but are not limited to i) adaptation improvement of advanced techniques in adversarial attacks and backdoor attacks; ii) introducing the interpretable methods for DNNs like attribution and hidden-semantic analysis [229]; iii) exploring the mathematics of the generalization ability of DNNs in depth like optimal transport or information geometry.

(4) Exploring More Functions about Deep IPs. As this survey has shown, most existing DNN watermarking and fingerprinting schemes focus on copyright verification. Several works focus on other functions like integrity verification, user management or applicability authorization. However, more functions are worth considering. For example, semi-fragile watermarks or fingerprints can be researched to allow legitimate users to modify the model within limits to support their own personalized applications. Here, more functions can be explored from the ideas of traditional data watermarking or the properties of DNNs like high-dimensional nonlinear decision boundary and generalization domain.

(5) Fair and Comprehensive Benchmark Toolkits for Deep IP Defenses and Attacks. In the existing research about DeepIP, the implementation and performance evaluation are conducted on their respective experimental settings, which would confuse the users of Deep IP schemes. Several works have deployed a toolkit or built a fair setting to evaluate the robustness of some classical methods. However, the metrics, tasks, and scenarios they consider are not comprehensive, and there is still a distance to mature commercial standardized platforms. Such benchmark toolkits will guide users to choose the appropriate solution to protect their models according to their actual needs.

(6) Deep IP for Large Foundation Models. AI applications based on large foundation models have covered various aspects of human daily lives, like question answering (ChatGPT[6]) and voice assistants. Since building a high-performance large model costs a lot on infrastructure construction, data collection and labelling, and model training, it is an urgent need to build a complete Deep IP tool for large models with strong attack resistance, acceptable overhead and imperceptible performance degradation. The following challenges should be considered: i) Large models have more complexity and uncertainty, and their behavioural boundaries are difficult to explore and interpret. Embedding watermarks will significantly degrade the performance and introduce agnostic security risks, while the fingerprints of large models are more difficult to extract and have few guarantees to retain in various attacks or task adaptations; ii) Large foundation models are usually cross-modal like GPT-4 [230] and DALL-E 2 [231] that support text and image data, and apply new model structures or learning techniques unexplored in the area of Deep IP like Denoising Diffusion Probabilistic Models and Transformer; iii) Once a high-performance large model is released, it will scale to various downstream tasks. Criminals will also use these powerful tools for malicious purposes. Thus, besides ensuring attack resistance, Deep IP tools should have functions to prevent or trace model abuse.

(7) Exploring Diverse IP Attack Methods and Targeted Defense Techniques. This survey has summarized some exsiting attacks to defeat Deep IPs, including three types: IP detection

6. https://openai.com/product

& evasion, IP removal and IP ambiguity. More vulnerabilities in the process of registering and verifying IP signatures need to be discovered, so that the defender can propose targeted defense strategies to avoid potential risks. Defence and attack methods will complement each other and develop together, which promotes the maturity of Deep IP protection.

(8) Automated Discovery for Potential Infringement Facts within low costs. To protect Deep IPs not only requires a scheme that meets the basic metrics, but also needs to accurately locate the MLaaS providers using the piracy model. It can take the form of ante-hoc prevention and post-hoc discovery. The former can be realized by identifying specific request patterns from model stealers, including request data patterns and network state patterns. For the latter, consider black-box and none-box verification scenarios. Verifiers in black-box scenarios submit service requests to suspect remote APIs, typically trigger samples predefined by model owners, and then extract IP signatures from the returned query results. However, there are a lot of MLaaS APIs on the web, and most MLaaS providers follow a pay-per-request business model. Therefore, it is critical to control the verification cost while minimizing the number of trigger samples while ensuring the verification accuracy. Moreover, automated discovery in none-box scenarios needs to design a mechanism to perceive and acquire the suspect data and its source suspect model, so as to complete the whole pipeline from prevention, perception to forensics.

(9) The Scalability to Various DNN Models and Tasks. At present, most works for Deep IPs are designed for basic classification models, while several works focus on other tasks such as self-supervised learning, generative models and transfer learning. However, Deep IP protection of most existing deep learning techniques remains open, which can be explored from the following views: i) new model structures like graph neural networks, denoising diffusion models, capsule networks, etc; ii) new learning paradigms like reinforcement learning, geometric deep learning, multimodal learning, etc; iii) New tasks like object detection frameworks, 3D reconstruction, SAR image processing, spatial-temporal deep learning, etc.

(10) Deep IP Schemes not relying on Algorithmic Concealment. Most existing Deep IP schemes are based on algorithmic concealment to realize the discriminability of IP signatures, which assumes that the adversary does not know the specific details of the algorithm. However, schemes based on this assumption have been proven to be unreliable in the long-term practice in the field of information security. The Kerckhoffs principle should be followed to develop a Deep IP solution that relies only on key security: Even if the Deep IP algorithm is known by the adversary, it is difficult for the adversary to remove or modify the constructed IP signatures of the target model on the premise of ensuring the model performance so as to avoid IP verification or falsely ownership claiming.

(11) Beyond Techniques: Standardization, Law and Policy. In addition to the technical progress, how to endow the techniques for Deep IP protection with legal effects is also worthy of focus. Firstly, the industry should form a standardized workflow for Deep IP protection, and only the models that comply with these standards will be protected. These standards should be sufficiently secure and evolve as potential vulnerabilities are discovered. Secondly, judicial institutions and government functional departments need to study and implement deterrent and binding laws and policies to combat the theft, piracy, abuse and other malicious behaviors of deep learning models, and promote the healthy development of deep learning and its related industries.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012, pp. 1097–1105.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[4] A. Jabbar, X. Li, and B. Omar, "A survey on generative adversarial networks: Variants, applications, and training," *ACM Computing Surveys*, vol. 54, no. 8, 2021.

[5] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *IJCV*, vol. 128, no. 2, pp. 261–318, 2020.

[6] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *CVPR*, 2014, pp. 1701–1708.

[7] J. Li *et al.*, "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022.

[8] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pretraining of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[9] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *ICCV*, 2015, pp. 2722–2730.

[10] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[11] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackerman *et al.*, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, pp. 688–702, 2020.

[12] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[13] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multiagent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[14] N. Brown and T. Sandholm, "Superhuman ai for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[15] A. Da'u and N. Salim, "Recommendation system based on deep learning methods: a systematic review and new directions," *Artificial Intelligence Review*, vol. 53, no. 4, pp. 2709–2748, 2020.

[16] L. Antonyshyn, J. Silveira, S. Givigi, and J. Marshall, "Multiple mobile robot task and motion planning: A survey," *ACM Computing Surveys*, 2022.

[17] Z. Sun, R. Sun, L. Lu, and A. Mislove, "Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1955–1972.

[18] D. Oliynyk, R. Mayer, and A. Rauber, "I know what you trained last summer: A survey on stealing machine learning models and defences," *arXiv preprint arXiv:2206.08451*, 2022.

[19] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (ICMR)*, 2017, pp. 269–277.

[20] 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 *Proceedings of the 2019 on International Conference on Multimedia Retrieval (ICMR)*, 2019, pp. 105–113.

[21] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: an end-to-end watermarking framework for protecting the ownership of deep neural networks," in *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019.

[22] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[23] L. Fan, K. W. Ng, C. S. Chan, and Q. Yang, "Deepipr: Deep neural network intellectual property protection with passports," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[24] E. Le Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications (NCA)*, vol. 32, no. 13, pp. 9233–9244, 2020.

[25] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1615–1631.

[26] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (AsiaCCS)*, 2018, pp. 159–172.

[27] H. Chen, B. D. Rouhani, and F. Koushanfar, "Blackmarks: Blackbox multibit watermarking for deep neural networks," *arXiv preprint arXiv:1904.00344*, 2019.

[28] X. Cao, J. Jia, and N. Z. Gong, "Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 14–25.

[29] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," 2020.

[30] X. Pan, M. Zhang, Y. Lu, and M. Yang, "Tafa: A task-agnostic fingerprinting algorithm for neural networks," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 542–562.

[31] Z. Peng, S. Li, G. Chen, C. Zhang, H. Zhu, and M. Xue, "Fingerprinting deep neural networks globally via universal adversarial perturbations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13 430–13 439.

[32] G. Liu, T. Xu, X. Ma, and C. Wang, "Your model trains on my data? protecting intellectual property of training data via membership fingerprint authentication," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1024–1037, 2022.

[33] X. Pan, Y. Yan, M. Zhang, and M. Yang, "Metav: A meta-verifier approach to task-agnostic model fingerprinting," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1327–1336.

[34] Y. Li, Z. Zhang, B. Liu, Z. Yang, and Y. Liu, "Modeldiff: testing-based dnn similarity comparison for model reuse detection," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 139–151.

[35] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song, "Copy, right? a testing framework for copyright protection of deep learning models," in *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[36] F. Regazzoni, P. Palmieri, F. Smailbegovic, R. Cammarota, and I. Polian, "Protecting artificial intelligence ips: a survey of watermarking and fingerprinting for machine learning," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 2, pp. 180–191, 2021.

[37] M. Xue, Y. Zhang, J. Wang, and W. Liu, "Intellectual property protection for deep learning models: Taxonomy, methods, attacks, and evaluations," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 01, pp. 1–1, 2022.

[38] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "Sok: How robust is image classification deep neural network watermarking?" *IEEE S&P*, 2022.

[39] F. Boenisch, "A systematic review on model watermarking for neural networks," *Frontiers in big Data*, vol. 4, 2021.

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

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

[42] A. Fkirin, G. Attiya, A. El-Sayed, and M. A. Shouman, "Copyright protection of deep neural network models using digital watermarking: a comparative study," *Multimedia Tools and Applications*, vol. 81, no. 11, pp. 15 961–15 975, 2022.

[43] M. Barni, F. Pérez-González, and B. Tondi, "Dnn watermarking: four challenges and a funeral," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 189–196.

[44] S. Peng, Y. Chen, J. Xu, Z. Chen, C. Wang, and X. Jia, "Intellectual property protection of dnn models," *World Wide Web*, pp. 1–35, 2022.

[45] S. Lee, W. Song, S. Jana, M. Cha, and S. Son, "Evaluating the robustness of trigger set-based watermarks embedded in deep neural networks," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2022.

[46] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1345–1362.

[47] Y. He, G. Meng, K. Chen, X. Hu, and J. He, "Towards security threats of deep learning systems: A survey," *IEEE Transactions on Software Engineering*, 2020.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[49] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[50] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[51] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2575–2584.

[52] L. Wang, S. Xu, R. Xu, X. Wang, and Q. Zhu, "Non-transferable learning: A new approach for model ownership verification and applicability authorization," in *International Conference on Learning Representations*, 2022.

[53] N. Lin, X. Chen, H. Lu, and X. Li, "Chaotic weights: A novel approach to protect intellectual property of deep neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 7, pp. 1327–1339, 2020.

[54] M. Alam, S. Saha, D. Mukhopadhyay, and S. Kundu, "Nn-lock: A lightweight authorization to prevent ip threats of deep learning models," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 3, pp. 1–19, 2022.

[55] M. Xue, Z. Wu, Y. Zhang, J. Wang, and W. Liu, "Advparams: An active dnn intellectual property protection technique via adversarial perturbation based parameter encryption," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[56] A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou, "Radioactive data: tracing through training," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8326–8335.

[57] W. Peng and J. Chen, "Learnability lock: Authorized learnability control through adversarial invertible transformations," in *International Conference on Learning Representations (ICLR)*, 2022.

[58] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," *arXiv preprint arXiv:1704.06857*, 2017.

[59] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3365–3387, 2020.

[60] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 11, pp. 3365–3385, 2019.

[61] S. Ranathunga, E.-S. A. Lee, M. Prifti Skenduli, R. Shekhar, M. Alam, and R. Kaur, "Neural machine translation for low-resource languages: A survey," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–37, 2023.

[62] T. Wang and F. Kerschbaum, "Attacks on digital watermarks for deep neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2622–2626.

[63] ——, "Riga: Covert and robust white-box watermarking of deep neural networks," in *Proceedings of the Web Conference 2021 (WWW)*, 2021, pp. 993–1004.

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

[65] H. Wang, M. Xue, S. Sun, Y. Zhang, J. Wang, and W. Liu, "Detect and remove watermark in deep neural networks via generative adversarial networks," in *International Conference on Information Security (ISC 2021)*, 2021.

[66] R. Namba and J. Sakuma, "Robust watermarking of neural network with exponential weighting," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2019, pp. 228–240.

[67] W. Aiken, H. Kim, S. Woo, and J. Ryoo, "Neural network laundering: Removing black-box backdoor watermarks from deep neural networks," *Computers & Security*, vol. 106, p. 102277, 2021.

[68] W.-A. Lin, Y. Balaji, P. Samangouei, and R. Chellappa, "Invert and defend: Model-based approximate inversion of generative adversarial networks for secure inference," *arXiv preprint arXiv:1911.10291*, 2019.

[69] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," in *International Conference on Learning Representations(ICLR)*, 2019.

[70] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," 2018.

[71] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, "Efficient defenses against adversarial attacks," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 39–49.

[72] 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 *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 21*, 2021, pp. 3635–3641.

[73] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

[74] R. Wang, H. Li, L. Mu, J. Ren, S. Guo, L. Liu, L. Fang, J. Chen, and L. Wang, "Rethinking the vulnerability of dnn watermarking: Are watermarks robust against naturalness-aware perturbations?" in *Proceedings of the 30th ACM International Conference on Multimedia (MM)*, 2022, pp. 1808–1818.

[75] X. Chen, W. Wang, Y. Ding, C. Bender, R. Jia, B. Li, and D. Song, "Leveraging unlabeled data for watermark removal of deep neural networks," in *ICML workshop on Security and Privacy of Machine Learning*, 2019, pp. 1–6.

[76] X. Chen, W. Wang, C. Bender, Y. Ding, R. Jia, B. Li, and D. Song, "Refit: a unified watermark removal framework for deep learning systems with limited data," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2021, pp. 321–335.

[77] X. Liu, F. Li, B. Wen, and Q. Li, "Removing backdoor-based watermarks in neural networks with limited data," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 10 149–10 156.

[78] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.

[79] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *Proceedings of machine learning and systems*, vol. 2, pp. 129–146, 2020.

[80] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.

[81] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[82] M. Shafieinejad, N. Lukas, J. Wang, X. Li, and F. Kerschbaum, "On the robustness of backdoor-based watermarking in deep neural networks," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 177–188.

[83] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni, F. Koushanfar, A.-R. Sadeghi, and T. Schneider, "FLAME: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[84] Z. Zhang, Y. Li, J. Wang, B. Liu, D. Li, Y. Guo, X. Chen, and Y. Liu, "Remos: reducing defect inheritance in transfer learning via relevant model slicing," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1856–1868.

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

[86] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[87] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[88] Q. Zhong, L. Y. Zhang, S. Hu, L. Gao, J. Zhang, and Y. Xiang, "Attention distraction: Watermark removal through continual learning with selective forgetting," in *ICME*, 2022.

[89] B. Goldberger, G. Katz, Y. Adi, and J. Keshet, "Minimal modifications of deep neural networks using verification," in *LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, vol. 73, 2020, pp. 260–278.

[90] Y. Xue, Y. Zhu, Z. Zhu, S. Li, Z. Qian, and X. Zhang, "Removing watermarks for image processing networks via referenced subspace attention," *The Computer Journal*, 2022.

[91] Y. Yan, X. Pan, M. Zhang, and M. Yang, "Rethinking white-box watermarks on deep learning models under neural structural obfuscation," in *32th USENIX security symposium (USENIX Security 23)*, 2023.

[92] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4954–4963.

[93] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

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

[95] L. Charette, L. Chu, Y. Chen, J. Pei, L. Wang, and Y. Zhang, "Cosine model watermarking against ensemble distillation," in *AAAI*, 2022.

[96] H. Liu, Z. Weng, and Y. Zhu, "Watermarking deep neural networks with greedy residuals," in *International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 6978–6988.

[97] L. Feng and X. Zhang, "Watermarking neural network with compensation mechanism," in *International Conference on Knowledge Science, Engineering and Management (KSEM)*. Springer, 2020, pp. 363–375.

[98] M. Kuribayashi, T. Tanaka, S. Suzuki, T. Yasui, and N. Funabiki, "White-box watermarking scheme for fully-connected layers in fine-tuning model," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec)*, 2021, pp. 165–170.

[99] X. Guan, H. Feng, W. Zhang, H. Zhou, J. Zhang, and N. Yu, "Reversible watermarking in deep convolutional neural networks for integrity authentication," in *Proceedings of the 28th ACM International Conference on Multimedia (MM)*, 2020, pp. 2273–2280.

[100] M. Botta, D. Cavagnino, and R. Esposito, "Neunac: A novel fragile watermarking algorithm for integrity protection of neural networks," *Information Sciences*, vol. 576, pp. 228–241, 2021.

[101] H. Chen, C. Fu, B. D. Rouhani, J. Zhao, and F. Koushanfar, "Deepattest: an end-to-end attestation framework for deep neural networks," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2019, pp. 487–498.

[102] Y. Li, L. Zhu, X. Jia, Y. Jiang, S.-T. Xia, and X. Cao, "Defending against model stealing via verifying embedded external features," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 36, no. 2, 2022, pp. 1464–1472.

[103] X. Zhao, Y. Yao, H. Wu, and X. Zhang, "Structural watermarking to deep neural networks via network channel pruning," in *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2021, pp. 1–6.

[104] C. Xie, P. Yi, B. Zhang, and F. Zou, "Deepmark: Embedding watermarks into deep neural network using pruning," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2021, pp. 169–175.

[105] X. Chen, T. Chen, Z. Zhang, and Z. Wang, "You are caught stealing my winning lottery ticket! making a lottery ticket claim its ownership," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1780–1791, 2021.

[106] X. Lou, S. Guo, T. Zhang, Y. Zhang, and Y. Liu, "When nas meets watermarking: ownership verification of dnn models via cache side channels," *TCSVT*, 2022.

[107] J. Zhang, D. Chen, J. Liao, W. Zhang, G. Hua, and N. Yu, "Passport-aware normalization for deep model protection," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 22 619–22 628, 2020.

[108] 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 Recognition*, vol. 122, p. 108285, 2022.

[109] P. Lv, P. Li, S. Zhang, K. Chen, R. Liang, H. Ma, Y. Zhao, and Y. Li, "A robustness-assured white-box watermark in neural networks," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[110] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[111] A. P. Maung Maung and H. Kiya, "Piracy-resistant dnn watermarking by block-wise image transformation with secret key," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security (IHMM&Sec)*, 2021, pp. 159–164.

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

[113] Y. Lao, P. Yang, W. Zhao, and P. Li, "Identification for deep neural network: Simply adjusting few weights!" in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1328–1341.

[114] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1937–1954.

[115] 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 *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, 2019, pp. 126–137.

[116] Y. Lao, W. Zhao, P. Yang, and P. Li, "Deepauth: A dnn authentication framework by model-unique and fragile signature embedding," in *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI), Virtual Event*, 2022.

[117] Y. Li, Y. Bai, Y. Jiang, Y. Yang, S.-T. Xia, and B. Li, "Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection," in *NeurIPS*, 2022.

[118] 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 *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2020, pp. 462–474.

[119] D. Mehta, N. Mondol, F. Farahmandi, and M. Tehranipoor, "Aime: watermarking ai models by leveraging errors," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 304–309.

[120] Y. Li, L. Zhu, X. Jia, Y. Bai, Y. Jiang, S.-T. Xia, and X. Cao, "Move: Effective and harmless ownership verification via embedded external features," *arXiv preprint arXiv:2208.02820*, 2022.

[121] P. Yang, Y. Lao, and P. Li, "Robust watermarking for deep neural networks via bi-level optimization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 841–14 850.

[122] A. Bansal, P.-Y. Chiang, M. J. Curry, R. Jain, C. Wigington, V. Manjunatha, J. P. Dickerson, and T. Goldstein, "Certified neural network watermarks with randomized smoothing," in *International Conference on Machine Learning (ICML)*. PMLR, 2022, pp. 1450–1465.

[123] G. Ren, J. Wu, G. Li, S. Li, and M. Guizani, "Protecting intellectual property with reliable availability of learning models in ai-based cybersecurity services," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2022.

[124] J. Zhang, D. Chen, J. Liao, H. Fang, W. Zhang, W. Zhou, H. Cui, and N. Yu, "Model watermarking for image prolearning multiple layers of features from tiny imagescessing networks," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, no. 07, 2020, pp. 12 805–12 812.

[125] J. Zhang, D. Chen, J. Liao, W. Zhang, H. Feng, G. Hua, and N. Yu, "Deep model intellectual property protection via deep watermarking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[126] H. Wu, G. Liu, Y. Yao, and X. Zhang, "Watermarking neural networks with watermarked images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 7, pp. 2591–2601, 2020.

[127] N. Yu, V. Skripniuk, S. Abdelnabi, and M. Fritz, "Artificial fingerprinting for generative models: Rooting deepfake attribution in training data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 448–14 457.

[128] S. Abdelnabi and M. Fritz, "Adversarial watermarking transformer: Towards tracing text provenance with data hiding," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 121–140.

[129] X. He, Q. Xu, L. Lyu, F. Wu, and C. Wang, "Protecting intellectual property of language generation apis with lexical watermark," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 10 758–10 766.

[130] X. He, Q. Xu, Y. Zeng, L. Lyu, F. Wu, J. Li, and R. Jia, "CATER: Intellectual property protection on text generation APIs via conditional watermarks," in *Advances in Neural Information Processing Systems*, 2022.

[131] J. Jia, Y. Wu, A. Li, S. Ma, and Y. Liu, "Subnetwork-lossless robust watermarking for hostile theft attacks in deep transfer learning models," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2022.

[132] B. G. Tekgul, Y. Xia, S. Marchal, and N. Asokan, "Waffle: Watermarking in federated learning," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2021, pp. 310–320.

[133] B. Li, L. Fan, H. Gu, J. Li, and Q. Yang, "Fedipr: Ownership verification for federated deep neural network models," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

[134] S. Shao, W. Yang, H. Gu, J. Lou, Z. Qin, L. Fan, Q. Yang, and K. Ren, "Fedtracker: Furnishing ownership verification and traceability for federated learning model," *arXiv preprint arXiv:2211.07160*, 2022.

[135] T. Wu, X. Li, Y. Miao, M. Xu, H. Zhang, X. Liu, and K.-K. R. Choo, "Cits-mew: Multi-party entangled watermark in cooperative intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[136] D. S. Ong, C. S. Chan, K. W. Ng, L. Fan, and Q. Yang, "Protecting intellectual property of generative adversarial networks from ambiguity attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3630–3639.

[137] T. Cong, X. He, and Y. Zhang, "Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022, pp. 579–593.

[138] L. Sofiane, M. Njeh, O. Ermis, M. Önen, and S. Trabelsi, "Yes we can: Watermarking machine learning models beyond classification," in *CFS 2021, 34th IEEE Computer Security Foundations Symposium*, 2021.

[139] Z. Sun, X. Du, F. Song, M. Ni, and L. Li, "Coprotector: Protect open-source code against unauthorized training usage with data poisoning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 652–660.

[140] H. Chen, B. D. Rouhani, and F. Koushanfar, "Specmark: A spectral watermarking framework for ip protection of speech recognition systems," in *INTERSPEECH*, 2020, pp. 2312–2316.

[141] S. Szyller and N. Asokan, "Conflicting interactions among protections mechanisms for machine learning models," 2023.

[142] Y. Nagai, Y. Uchida, S. Sakazawa, and S. Satoh, "Digital watermarking for deep neural networks," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 1, pp. 3–16, 2018.

[143] G. Zhao, C. Qin, H. Yao, and Y. Han, "Dnn self-embedding watermarking: Towards tampering detection and parameter recovery for deep neural network," *Pattern Recognition Letters*, vol. 164, pp. 16–22, 2022.

[144] P. Maini, M. Yaghini, and N. Papernot, "Dataset inference: Ownership resolution in machine learning," in *ICLR*, 2021.

[145] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations 2019*.

[146] R. Zhu, P. Wei, S. Li, Z. Yin, X. Zhang, and Z. Qian, "Fragile neural network watermarking with trigger image set," in *International Conference on Knowledge Science, Engineering and Management*. Springer, 2021, pp. 280–293.

[147] L. Lin and H. Wu, "Verifying integrity of deep ensemble models by lossless black-box watermarking with sensitive samples," in *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE, 2022, pp. 1–6.

[148] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[149] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.

[150] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "Dawn: Dynamic adversarial watermarking of neural networks," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 4417–4425.

[151] J. Fei, Z. Xia, B. Tondi, and M. Barni, "Supervised gan watermarking for intellectual property protection," in *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2022, pp. 1–6.

[152] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," *arXiv preprint arXiv:2301.10226*, 2023.

[153] T. Xiang, C. Xie, S. Guo, J. Li, and T. Zhang, "Protecting your nlg models with semantic and robust watermarks," *arXiv preprint arXiv:2112.05428*, 2021.

[154] X. Zhao, L. Li, and Y.-X. Wang, "Distillation-resistant watermarking for model protection in nlp," *arXiv preprint arXiv:2210.03312*, 2022.

[155] X. Zhao, Y.-X. Wang, and L. Li, "Protecting language generation models via invisible watermarking," *arXiv preprint arXio:2302.03162*, 2023.

[156] A. Dziedzic, N. Dhawan, M. A. Kaleem, J. Guan, and N. Papernot, "On the difficulty of defending self-supervised learning against model extraction," 2022.

[157] T. Zhang, H. Wu, X. Lu, and G. Sun, "Awencoder: Adversarial watermarking pre-trained encoders in contrastive learning," *arXiv preprint arXiv:2208.03948*, 2022.

[158] M. AprilPyone and H. Kiya, "Transfer learning-based model protection with secret key," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3877–3881.

[159] Q. Yang, A. Huang, L. Fan, C. S. Chan, J. H. Lim, K. W. Ng, D. S. Ong, and B. Li, "Federated learning with privacy-preserving and model ip-right-protection," *Machine Intelligence Research*, vol. 20, no. 1, pp. 19–37, 2023.

[160] F.-Q. Li, S.-L. Wang, and A. W.-C. Liew, "Watermarking protocol for deep neural network ownership regulation in federated learning," in *2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 2022, pp. 1–4.

[161] ——, "Regulating ownership verification for deep neural networks: Scenarios, protocols, and prospects," *IJCAI 2021 Workshop on Toward IPR on Deep Learning as Services*, 2021.

[162] X. Liu, S. Shao, Y. Yang, K. Wu, W. Yang, and H. Fang, "Secure federated learning model verification: A client-side backdoor triggered watermarking scheme," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2021, pp. 2414–2419.

[163] T. Qiao, Y. Ma, N. Zheng, H. Wu, Y. Chen, M. Xu, and X. Luo, "A novel model watermarking for protecting generative adversarial network," *Computers & Security*, p. 103102, 2023.

[164] H. Ruan, H. Song, B. Liu, Y. Cheng, and Q. Liu, "Intellectual property protection for deep semantic segmentation models," *Frontiers of Computer Science*, vol. 17, no. 1, p. 171306, 2023.

[165] Y. Quan, H. Teng, Y. Chen, and H. Ji, "Watermarking deep neural networks in image processing," *IEEE Transactions on neural networks and learning systems*, vol. 32, no. 5, pp. 1852–1865, 2020.

[166] H. Jia, M. Yaghini, C. A. Choquette-Choo, N. Dullerud, A. Thudi, V. Chandrasekaran, and N. Papernot, "Proof-of-learning: Definitions and practice," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1039–1056.

[167] Y. Zheng, S. Wang, and C.-H. Chang, "A dnn fingerprint for non-repudiable model ownership identification and piracy detection," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2977–2989, 2022.

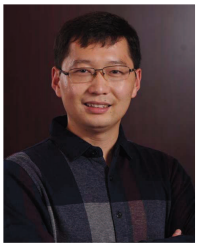
[168] C. Xiong, G. Feng, X. Li, X. Zhang, and C. Qin, "Neural network model protection with piracy identification and tampering localization capability," in *Proceedings of the 30th ACM International Conference on Multimedia (MM)*, 2022, pp. 2881–2889.
[169] H. Chen, H. Zhou, J. Zhang, D. Chen, W. Zhang, K. Chen, G. Hua, and N. Yu, "Perceptual hashing of deep convolutional neural networks for model copy detection," *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, 2022.
[170] J. Zhao, Q. Hu, G. Liu, X. Ma, F. Chen, and M. M. Hassan, "Afa: Adversarial fingerprinting authentication for deep neural networks," *Computer Communications*, vol. 150, pp. 488–497, 2020.
[171] S. Wang, X. Wang, P.-Y. Chen, P. Zhao, and X. Lin, "Characteristic examples: High-robustness, low-transferability fingerprinting of neural networks." in *IJCAI*, 2021, pp. 575–582.
[172] K. Yang, R. Wang, and L. Wang, "Metafinger: Fingerprinting the deep neural networks with meta-training," in *IJCAI 2022*.
[173] T. Dong, H. Qiu, T. Zhang, J. Li, H. Li, and J. Lu, "Fingerprinting multi-exit deep neural network models via inference time," *arXiv preprint arXiv:2110.03175*, 2021.
[174] Z. He, T. Zhang, and R. Lee, "Sensitive-sample fingerprinting of deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4729–4737.
[175] S. Wang, S. Abuadbba, S. Agarwal, K. Moore, R. Sun, M. Xue, S. Nepal, S. Camtepe, and S. Kanhere, "Publiccheck: Public integrity verification for services of run-time deep models," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1239–1256.
[176] S. Wang, P. Zhao, X. Wang, S. Chin, T. Wahl, Y. Fei, Q. A. Chen, and X. Lin, "Intrinsic examples: Robust fingerprinting of deep neural networks," in *British Machine Vision Conference (BMVC)*, 2021.
[177] S. Wang and C.-H. Chang, "Fingerprinting deep neural networks-a deepfool approach," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.
[178] G. Li, G. Xu, H. Qiu, S. Guo, R. Wang, J. Li, and T. Zhang, "A novel verifiable fingerprinting scheme for generative adversarial networks," *arXiv preprint arXiv:2106.11760*, 2021.
[179] J. Guan, J. Liang, and R. He, "Are you stealing my model? sample correlation for fingerprinting deep neural networks," in *Advances in Neural Information Processing Systems*, 2022.
[180] A. Dziedzic, H. Duan, M. A. Kaleem, N. Dhawan, J. Guan, Y. Cattan, F. Boenisch, and N. Papernot, "Dataset inference for self-supervised models," in *Advances in Neural Information Processing Systems*, 2022.
[181] Y. Chen, C. Shen, C. Wang, and Y. Zhang, "Teacher model fingerprinting attacks against transfer learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3593–3610.
[182] H. Jia, H. Chen, J. Guan, A. S. Shamsabadi, and N. Papernot, "A zest of LIME: Towards architecture-independent model distances," in *International Conference on Learning Representations (ICLR)*, 2022.
[183] Y. Jeong, D. Kim, Y. Ro, P. Kim, and J. Choi, "Fingerprintnet: Synthesized fingerprints for generated image detection," in *European Conference on Computer Vision*. Springer, 2022, pp. 76–94.
[184] Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang, and A. Yuille, "Learning transferable adversarial examples via ghost networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 458–11 465.
[185] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.
[186] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.
[187] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE symposium on security and privacy (S&P)*. IEEE, 2017, pp. 39–57.
[188] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.
[189] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE transactions on software engineering*, vol. 37, no. 5, pp. 649–678, 2010.
[190] S. Demir, H. F. Eniser, and A. Sen, "Deepsmartfuzzer: Reward guided test generation for deep learning," *arXiv preprint arXiv:1911.10621*, 2019.
[191] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 16 000–16 009.
[192] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *European Conference on Computer Vision (ECCV)*. Springer, 2008, pp. 705–718.
[193] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
[194] J. Vonderfecht and F. Liu, "Fingerprints of super resolution networks," *Transactions on Machine Learning Research*, 2022.
[195] https://github.com/yu4u/dnn-watermark.
[196] https://github.com/dnn-security/Watermark-Robustness-Toolbox.
[197] https://github.com/eil/greedy-residuals.
[198] https://github.com/TIANHAO-WANG/riga.
[199] https://github.com/THUYimingLi/MOVE.
[200] https://github.com/lvpeizhuo/HufuNet.
[201] https://github.com/WSP-LAB/wm-eval-zoo.
[202] https://github.com/arpitbansal297/Certified_Watermarks.
[203] https://github.com/THUYimingLi/Untargeted_Backdoor_Watermark.
[204] https://github.com/zhenglisec/Blind-Watermark-for-DNN.
[205] https://github.com/cleverhans-lab/entangled-watermark.
[206] https://github.com/dunky11/adversarial-frontier-stitching.
[207] https://github.com/conditionWang/NTL.
[208] https://developer.huaweicloud.com/develop/aigallery/notebook/detail?id=2d937a91-1692-4f88-94ca-82e1ae8d4d79.
[209] https://github.com/kamwoh/DeepIPR.
[210] https://github.com/ZJZAC/Passport-aware-Normalization.
[211] https://github.com/dingsheng-ong/ipr-gan.
[212] https://github.com/jianhanlim/ipr-image-captioning.
[213] https://github.com/VITA-Group/NO-stealing-LTH.
[214] https://github.com/ssg-research/WAFFLE.
[215] https://github.com/purp1eHaze/FedIPR.
[216] https://github.com/tianshuocong/SSLGuard.
[217] https://github.com/v587su/CoProtector.
[218] https://github.com/cleverhans-lab/Proof-of-Learning.
[219] https://github.com/Testing4AI/DeepJudge.
[220] https://github.com/ayberkuckun/DNN-Fingerprinting.
[221] http://www.chenwang.net.cn/code/MeFA-Code.zip.
[222] https://github.com/kangyangWHU/MetaFinger/.
[223] https://github.com/ylimit/ModelDiff.
[224] https://github.com/guanjiyang/SAC.
[225] https://github.com/cleverhans-lab/Zest-Model-Distance.
[226] https://github.com/yfchen1994/Teacher-Fingerprinting.
[227] https://github.com/cleverhans-lab/dataset-inference.
[228] https://github.com/cleverhans-lab/DatasetInferenceForSelfSupervisedModels.
[229] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021.
[230] OpenAI, "Gpt-4 technical report," 2023.
[231] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

Yuchen Sun received the B.S. degree in Telecommunication Engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2018. He has been with the School of System Engineering, National University of Defense Technology (NUDT), Changsha, since 2018, where he is currently a PH.D candidate. His research interests include Trustworthy Artificial Intelligence, Distributed Systems, and Wireless Indoor Localization.

Tianpeng Liu Tianpeng Liu received the B.Eng. and M.Eng and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2008, 2011 and 2016 respectively. He is currently an associate professor of the College of Electronic Science and Technology, NUDT. His primary research interests are radar signal processing, automatic target recognition, and cross-eye jamming.

Panhe Hu Panhe Hu was born in Shandong, China, in 1991. He received his BS degree from the Xidian University, Xi'an, China, in 2013, and the PhD degree from National University of Defense Technology(NUDT), Changsha, China, in 2019. He is currently an associate professor with the College of Electronic Science and Technology, NUDT. His current research interests include radar signal design, array signal processing and deep learning.

Qing Liao received her Ph.D. degree in computer science and engineering in 2016 supervised by Prof. Qian Zhang from the Department of Computer Science and Engineering of the Hong Kong University of Science and Technology. She is currently a professor with School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. Her research interests include artificial intelligence, data mining and information security.



Shouling Ji Shouling Ji is a ZJU 100-Young Professor in the College of Computer Science and Technology at Zhejiang University and a Research Faculty in the School of Electrical and Computer Engineering at Georgia Institute of Technology (Georgia Tech). He received a Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology , a Ph.D. degree in Computer Science from Georgia State University, and B.S. (with Honors) and M.S. degrees both in Computer Science from Heilongjiang University. His current research interests include Data-driven Security and Privacy, AI Security and Big Data Analytics. He is a member of ACM, IEEE, and CCF and was the Membership Chair of the IEEE Student Branch at Georgia State University (2012-2013).

Nenghai Yu Nenghai Yu received the B.S. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1987, the M.E. degree from Tsinghua University, Beijing, China, in 1992, and the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China, in 2004. Since 1992, he has been a Faculty Member with the Department of Electronic Engineering and Information Science, USTC, where he is currently a Professor. He is the Executive Director of the Department of Electronic Engineering and Information Science, USTC, where he is the Director of the Information Processing Center. He has authored or coauthored over 130 papers in journals and international conferences. His current research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.

Deke Guo received the B.S. degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a Professor with the College of System Engineering, National University of Defense Technology, and is also with the College of Intelligence and Computing, Tianjin University. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM.

Li Liu received the Ph.D. degree in information and communication engineering from the National University of Defense Technology (NUDT), China, in 2012. She is currently a Full Professor with NUDT. During her Ph.D. study, she spent more than two years as a Visiting Student at the University of Waterloo, Canada, from 2008 to 2010. From 2015 to 2016, she spent ten months visiting the Multimedia Laboratory at the Chinese University of Hong Kong. From 2016.12 to 2018.11, she worked as a senior researcher at the Machine Vision Group at the University of Oulu, Finland. Her current research interests include Computer Vision, Machine Learning, Artificial Intelligence, Trustworthy AI, Synthetic Aperture Radar. Her papers have currently over 9500+ citations in Google Scholar.