machine learning & knowledge extraction

MDPI

Review

When Federated Learning Meets Watermarking: A Comprehensive Overview of Techniques for Intellectual Property Protection

Mohammed Lansari [1,2,†,*], Reda Bellafqira [1,†], Katarzyna Kapusta [2], Vincent Thouvenot [2], Olivier Bettan [2] and Gouenou Coatrieux [1]

[1] IMT Atlantique, Inserm UMR 1101, 29200 Brest, France; reda.bellafqira@imt-atlantique.fr (R.B.); gouenou.coatrieux@imt-atlantique.fr (G.C.)
[2] ThereSIS, Thales SIX GTS, 91120 Palaiseau, France; katarzyna.kapusta@thalesgroup.com (K.K.); vincent.thouvenot@thalesgroup.com (V.T.); olivier.bettan@thalesgroup.com (O.B.)
* Correspondence: mohammed.lansari@imt-atlantique.fr
† These authors contributed equally to this work.

Abstract: Federated learning (FL) is a technique that allows multiple participants to collaboratively train a Deep Neural Network (DNN) without the need to centralize their data. Among other advantages, it comes with privacy-preserving properties, making it attractive for application in sensitive contexts, such as health care or the military. Although the data are not explicitly exchanged, the training procedure requires sharing information about participants' models. This makes the individual models vulnerable to theft or unauthorized distribution by malicious actors. To address the issue of ownership rights protection in the context of machine learning (ML), DNN watermarking methods have been developed during the last five years. Most existing works have focused on watermarking in a centralized manner, but only a few methods have been designed for FL and its unique constraints. In this paper, we provide an overview of recent advancements in federated learning watermarking, shedding light on the new challenges and opportunities that arise in this field.

Keywords: DNN watermarking; federated learning; intellectual property; security

Citation: Lansari, M.; Bellafqira, R.; Kapusta, K.; Thouvenot, V.; Bettan, O.; Coatrieux, G. When Federated Learning Meets Watermarking: A Comprehensive Overview of Techniques for Intellectual Property Protection. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1382–1406. https://doi.org/10.3390/make5040070

Academic Editor: Simon Tjoa

Received: 17 August 2023
Revised: 29 September 2023
Accepted: 2 October 2023
Published: 4 October 2023

1. Introduction

In the fast-paced digital era, machine learning (ML) has emerged as a revolutionary technology that drives innovation across various industries. Its ability to analyze vast amounts of data and make accurate predictions has opened up a world of possibilities. From enabling personalized recommendations in e-commerce [1] and improving medical diagnoses [2] to enhancing autonomous vehicles [3], the applications of machine learning are both diverse and impactful.

The effectiveness of machine learning models depends on the availability of large, high-quality datasets. Traditionally, the success of ML algorithms relied on centralized data storage and processing, where organizations and institutions accumulated vast amounts of information to train their models effectively. Unfortunately, this approach increases the risk of privacy leakage, as sensitive information about individuals, customers, or proprietary processes may inadvertently be exposed to unauthorized entities. This presents a roadblock to the advancement of machine learning, as data holders are understandably hesitant to compromise their users' privacy and organizational security. Moreover, safeguarding the privacy and security of sensitive data is not just an ethical obligation but a legal necessity, as highlighted by various data protection regulations around the world, such as the General Data Protection Regulation (GDPR) [4] in the EU, the California Consumer Privacy Act (CCPA) [5] in the US, and Data Security Law (DSL) [6] in China.

Addressing these challenges, federated learning (FL) [7] emerges as a promising solution. FL enables multiple data owners or distributed agents to collectively train a

global model without directly sharing their private data. Incorporating privacy-preserving techniques like homomorphic encryption [8], multi-party computation [9], private set intersection (PSI) [10], differential privacy [11,12], or hardware-based solutions such as trusted execution environment (TEE) [13], FL ensures data privacy and security while enabling collaborative model training. The significance of federated learning lies not only in its ability to alleviate privacy concerns but also in its potential to unlock the untapped value of distributed data. By enabling cooperation among organizations and institutions, federated learning allows for more comprehensive and representative models, ultimately leading to more accurate predictions and valuable insights [14].

Another critical aspect to take into account is the protection of the intellectual property of the trained model, which is aimed at preventing theft, plagiarism, and unauthorized usage by external parties. In response to this concern, model watermarking has been introduced since 2017 [15] as a solution to embed a unique watermark or fingerprint into the model. This measure effectively safeguards the model's intellectual property and enables tracing the source in case of any illicit model leakage. However, the majority of model watermarking methods address the problem of local or centralized training. The collaborative setting of federated learning raises new challenges in the context of ownership protection. How to identify that a participant contributed to the training of a model resulting from federated learning? How to be sure that the final model will not be misused by the aggregation party that coordinates the federated learning? These issues may slow down parties from joining the learning consortium. Previously, Yang et al. [16] conducted a survey on federated watermarking solutions, primarily focusing on security issues within federated learning. However, the paper only presents two watermarking algorithms (Waffle [17] and FEDIPR [18]) specifically designed for the context of FL.

1.1. Contributions

This work aims to provide a comprehensive analysis of watermarking in the context of federated learning with benefits regarding an overview of recent advances and shedding light on the remaining challenges and opportunities in this field. Concisely, our paper makes the following main contributions:

- We emphasize the foundational principle of secure federated learning and stress the importance of comprehensive security guarantees that align with the trust levels of the participants. This includes addressing various aspects such as training on non-Independently and Identically Distributed (non-I.I.D) data and ensuring robustness against poisoning and backdooring attacks.
- We conduct a comprehensive examination of the challenges concerning model watermarking in the context of federated learning.
- We expand upon the work presented in [16] by providing an in-depth analysis of nine currently available DNN watermarking schemes in the FL context along with their limitations.
- We bring attention to unresolved issues that necessitate further exploration.

1.2. Organization

The rest of the paper is organized as follows. Section 2 describes background information on federated learning and the DNN watermarking concept. Section 3 presents our definition of watermarking for federated learning followed by an overview of the six existing works combining FL and DNN. In Section 4, we analyze then the different elements that have to be taken into consideration while designing a watermarking scheme for a collaborative training setting. We identify the unsolved challenges and thus give an insight into possible future works. Section 6 concludes the paper.

2. Background

In this section, we provide the definition of federated learning and give a brief overview of its different existing settings. Then, we introduce DNN watermarking.

2.1. Federated Learning

Federated learning is a machine learning setting where $K \in \mathbb{N}^*$ entities called clients collaborate to train a global model M_G while keeping the training data $D_C = \{D_{C_i}\}_{i=1}^{K}$ decentralized [19].

The most popular framework is called client-server FL or the scatter and gather framework. Here is a high-level overview of this setting:

1. Setup:
   - Central Server: A central server manages the overall training process and initiates model updates.
   - Clients: These are the individual devices, such as smartphones, IoT devices, or computers, that participate in the training process. Each edge device has its local dataset that cannot be shared with the central server due to privacy concerns.
2. Initialization:
   - Initially, the central server initiates a global model M_G (e.g., with random parameters) and distributes it to a subset of clients selected randomly at each round.
3. Local Training:
   - Each client trains the global model M_G on its local dataset using its computational resources. The training is typically performed using gradient descent or a similar optimization algorithm.
4. Model Update:
   - After the local training is complete, the clients generate a model update (typically gradients) based on the locally processed data.
5. Aggregation:
   - The clients send their model updates back to the central server without sharing their raw data.
   - The central server aggregates all received model updates to create a refined global model. This aggregation procedure can be completed in different ways [20–24]. To give an idea, the most common one is FedAvg [25] where the aggregation is completed by averaging the model weights (see Section 4.2).
6. Iterative Process:
   - Steps 3 to 5 are repeated for multiple rounds or epochs, allowing the global model to improve over time by leveraging knowledge from various clients.
7. Centralized Model Deployment:
   - Once the federated training process is complete, the final global model can be deployed from the central server to all clients for local inference.

Note that the presence of the server is not mandatory to perform FL. In a decentralized setting also known as a cyclic framework, clients can perform FL without the supervision of a server. BrainTorrent [26] proposes a server-less and peer-to-peer federated framework. In this solution, a random client is selected to be the aggregator. Then, it checks if other clients have an updated version of the model. If yes, they send it to the aggregator, who performs an averaging of the model weights/updates. Then, it updates its own model with the previous result.

Another type of decentralized learning is split learning [27], which consists of splitting the DNN model between the server and the clients. Many configurations are possible, but the most common for client privacy is the U-shaped configuration. The aim is that each client has the first and last layers of the DNN model and the server has the rest of the layers. In such a way, clients perform the forward/backward pass, keeping their data (inputs and labels) private, and send/receive only the activation maps/gradients, respectively, to update the whole model.

The FL is also defined by the data features partitioning, which can be categorized into three types: horizontal, vertical, and hybrid FL. In the case of horizontal FL, all clients share

the same set of features, but their sample data may differ. In contrast, vertical FL assumes that all clients possess the same sample data but have distinct features. Lastly, hybrid FL involves clients with varying samples and features across the board.

Ideally, the data should be Independently and Identically Distributed (I.I.D) for each client. However, in real-world FL scenarios, data distribution and the amount differ between clients since they collect and use their own data. This leads to a non-I.I.D data partition, which can result in significant performance loss [28]. To address this issue, several aggregation functions have been proposed, such as FedProx [20] and SCAFFOLD [22]. These methods aim to improve the performance of federated learning despite the non-I.I.D data distribution across clients.

The aim of federated learning is to ensure the security and privacy of clients' data while achieving a model's performance equivalent to centralized training. However, this objective can be compromised if the server or certain clients act maliciously. In real scenarios, the server may be considered an honest but curious entity, meaning it will abide by the FL protocol regarding client selection and global model aggregation/distribution, but it will attempt to infer information about the client data. This situation commonly arises in classical federated learning setups, where only model parameters/gradients are shared. The server can potentially employ attacks like the inversion attack, which reconstructs the data used for learning from the gradients shared by the clients. To counteract this type of attack, various countermeasures have been developed, one of which is homomorphic encryption (HE) [8]. Homomorphic encryption enables linear operations to be performed on data from their encrypted form (without having access to the secret key), preventing the server from conducting inversion attacks on encrypted gradients. However, using HE has some drawbacks, including high computational and communication complexity, and it also restricts the range of aggregation methods, as it only allows linear operations to be performed on the encrypted data.

Another approach to address privacy concerns is differential privacy, which combats inversion attacks by introducing noise to the updated gradients [11,12]. Nevertheless, this technique faces a limitation due to the trade-off between utility and privacy. Alternatively, trusted execution environments (TEEs) [13,29] present another solution, where a trusted third party is utilized to provide guarantees for code and data confidentiality and integrity. Multi-party computation (MPC) has also been implemented to achieve secure model aggregation in federated learning, as demonstrated in [13,29,30]. However, similar to encryption-based solutions, MPC-based approaches are susceptible to efficiency issues. In addition to reinforcing the confidentiality of the aggregated data, MPC—and more precisely its private set intersection protocol [10]—can be applied to identify the intersection of feature spaces between clients in vertically partitioned data.

In scenarios where a group of clients consists of malicious users, unlike the server, these clients have access to their local data and models. Leveraging this knowledge, they can employ various attacks to undermine the model's performance. Some of these attacks include poisoning attacks [31,32], which aim to introduce malicious data to corrupt the model's training or attacks that cause the model to misclassify data with specific patterns while maintaining its performance on the primary task, which is known as backdooring attacks [33,34]. These malicious actions pose significant challenges to the security and integrity of the federated learning process.

To defend against poisoning and backdooring attacks in the context of federated learning, two classes of approaches have been proposed. The first class involves developing robust aggregation techniques, such as Krum aggregation [23] or median and trimmed mean aggregation [35], which are designed to identify and remove abnormal local models contributed by potentially malicious clients. These techniques help improve the overall model's resilience to attacks. The second class of defenses relies on the use of anomaly detection techniques implemented by the server at each round of federated learning. These anomaly detection methods enable the server to identify and filter out abnormal client updates before performing the model aggregation process [36–38]. By doing so, the server

can mitigate the impact of potential malicious clients and enhance the security of the federated learning system. For more comprehensive insights into the threats and defenses related to federated learning, interested readers can refer to the work cited in [32].

To summarize, to establish a secure federated learning (FL) environment, the first step is to conduct a security analysis of the entities engaged in FL and assess their level of trust. This analysis helps identify potential threats and risks specific to the scenario being considered. Once the security analysis is complete, the next step involves integrating the security tools discussed earlier to design a robust FL model that performs effectively while ensuring data and model privacy. In Table 1, we present a list of secure federated learning frameworks along with the corresponding security tools that are employed to provide the necessary security measures. Furthermore, when developing an efficient and practical watermarking solution for federated learning, it is crucial to consider all the aforementioned security requirements. We elaborate on how these constraints can be accommodated in the context of federated learning watermarking in Section 4.

Table 1. Examples of open-source federated learning (FL) frameworks.

FL Frameworks	Developed by	Purpose	Security Protocols Provided
Fed-BioMed [39]	INRIA	Research	DP, HE
TensorFlow Federated [40]	Google	Research	DP
PySyft [41]	OpenMined	Research	MPC, DP, HE, PSI
Flower [42]	Flower Labs GmbH	Industrial	DP
FATE [43]	WeBank	Industrial	HE, DP, MPC
OpenFL [44]	Intel	Industrial	TEE
IBM Federated Learning [45]	IBM	Industrial	DP, MPC
NvFlare [46]	Nvidia	Industrial	HE, DP, PSI
Clara [47]	Nvidia	Industrial	DP, HE, TEE

2.2. DNN Watermarking

2.2.1. Requirements

DNN watermarking is a promising solution for ownership protection of ML models [48–53]. Inspired by multimedia and database watermarking [54–58], it consists of introducing a secret change into the model parameters or behavior during its training in order to enable its identification in the future. As multimedia content watermarking, DNN watermarking must respect some requirements to be effective for IP protection. Table 2 summarizes these requirements.

The watermark must be secret (Secrecy). This requirement refers to the fact that any person who analyzes the model is not able to detect if the latter is watermarked. White-Box techniques can change the distribution of the parameters and then differ from a non-watermarked model. In addition, a watermarking technique must also preserve the model's performance on the main task (Fidelity). If the watermarking embedding process returns a model that has an accuracy up to a defined ϵ compared to a non-watermarked model, then the watermarking technique is not efficient. Reliability ensures a low false negative rate for the owner during IP verification, while Integrity aims to prevent false positive claims by other parties. The watermarking algorithm should be independent of the model (Generality) while providing a large insertion Capacity, which can be either zero-bit, indicating only the presence of a watermark, or multi-bit, allowing the encoding of multiple bits of information.

Table 2. DNN watermarking requirements and properties.

Property	Description
Fidelity	The watermarked model needs to have performances as close as possible compared to the model without watermark.
Capacity	The capacity of a technique to embed multiple watermarks.
Reliability	Demonstrate a low false negative rate, enabling legitimate users to accurately identify their intellectual property with a high level of certainty.
Integrity	Demonstrate a low false positive rate, preventing unjustly accusing honest parties with similar models of intellectual property theft.
Generality	The capacity of a watermarking technique to be applied independently of the architecture of the model.
Efficiency	The performance cost generated by the embedding and verification process of the watermarking.
Robustness	The capacity to resist against attacks aiming at removing the watermark.
Secrecy	The watermark should be secret and undetectable.

Furthermore, the watermark must exhibit Robustness against attacks aimed at removing or detecting it. A removal attack is considered effective if it maintains a high test accuracy while eliminating the watermark. It is efficient if the resources required for the attack, such as runtime, are relatively small compared to retraining the model from scratch. Some common attacks include the following:

- Pruning Attack: Setting the less useful weights of the model to zero.
- Fine-Tuning Attack: Re-training the model and updating its weights without decreasing accuracy.
- Overwriting Attack: Embedding a new watermark to replace the original one.
- Wang and Kerschbaum Attack: For static White-Box watermarking algorithms, it alters the weight distribution of the watermarked model, relying on visual inspection.
- Property Inference Attack: Training a discriminating model to distinguish watermarked from non-watermarked models, thereby detecting if a protected model is no longer watermarked.
- Another attack is the Ambiguity Attack, which forges a new watermark on a model, making it challenging for external entities, like legal authorities, to determine the legitimate watermark owner. This ambiguity prevents the legitimate owner from claiming the copyright of the intellectual property.

2.2.2. Related Works

DNN watermarking can be distinguished into two types of techniques: White-Box [15,59–64,64–71] and Black-Box [65,72–85] watermarking. Each technique is defined by the type of access to the model parameters during the verification process.

In the White-Box setting, we assume that the owner will have full access to the model (architecture, parameters, activation maps...). In this way, to insert a watermark into a DNN, the owner will hide a piece of information b in the form of a binary string or an image (e.g., Quick Response code) into the model's parameters [15,71,86], activation maps [65,67,68] or by adding a passport layer [66,87]. As formulated in [68], a White-Box watermarking scheme is defined as follows:

1. Initially, a target model M is considered, and a features extraction function $Ext(M, K_{ext})$ is applied with a secret key K_{ext}. The features obtained can be a subset of the model weights, where K_{ext} indicates the indices of the selected weights. Alternatively, the features can be model activation maps for specific input data secretly chosen from a trigger set. These features are then utilized for watermark insertion and extraction.
2. The embedding of a watermark message b involves regularizing M using a specific regularization term E_{wat}. This regularization term ensures that the projection function $Proj(., K_{Proj})$ applied to the selected features encodes the watermark b in a predetermined watermark space, which depends on the secret key K_{Proj}. The goal is to achieve the following after training:

$$Proj(Ext(M^{wat}, K_{ext}), K_{Proj}) = b \tag{1}$$

where M^{wat} is the watermarked version of the target model M. To achieve this, the watermarking regularization term E_{wat} relies on a distance measure d defined in the watermark space. For example, in the case of a binary watermark with a binary string of length l, i.e., $b \in \{0,1\}^l$, the distance measure could be the Hamming distance, Hinge distance, or Cross-Entropy. The regularization term is formulated as follows:

$$E_{wat} = d(Proj(Ext(M^{wat}, K_{ext}), K_{Proj}), b) \tag{2}$$

To preserve the accuracy of the target model, the watermarked model M^{wat} is usually derived from M through a fine-tuning operation parameterized with the following loss function:

$$E = E_0 + \lambda E_{wat} \tag{3}$$

where E_0 represents the original loss function of the network, which is essential to ensure good performance in the main task. E_{wat} is the regularization term added to facilitate proper watermark extraction, and λ is a parameter that adjusts the trade-off between the original loss term and the regularization term.
3. The watermark retrieval process is relatively straightforward. It involves using both the features extraction function $Ext(., K_{ext})$ and the projection function $Proj(., K_{Proj})$ as follows:

$$b^{ext} = Proj(Ext(M^{wat}, K_{ext}), K_{Proj}) \tag{4}$$

where b^{ext} is the extracted message from the watermarked model M^{wat}.

For example, in the watermarking scheme introduced by Uchida et al. [15], the feature extraction function $Ext(., K_{ext})$ involves computing the mean value of secretly selected filter weights $\mathbf{w}$, where K_{ext} represents the index of the chosen convolutional layer. The projection function $Proj(., K_{Proj})$ in [15] is designed to insert a watermark $b \in \{0,1\}^l$, where l is the length of the watermark, and it is defined as follows:

$$Proj(\mathbf{w}, K_{Proj}) = \sigma(\mathbf{w} K_{Proj}) \in \{0,1\}^l \tag{5}$$

Here, K_{Proj} represents a secret random matrix of size $(|w|, l)$, and $\sigma(.)$ is the Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Uchida et al. [15] use binary cross-entropy as the distance measure d for the watermarking regularization E_{wat}, which is given by:

$$d(b, y) = -\sum_{j=1}^{l} (b_j \log(y_j) + (1 - b_j) \log(1 - y_j)) \tag{6}$$

With the information on d, $Proj(., K_{Proj})$, and $Ext(., K_{ext})$, the loss E_{wat} for watermarking a target model M can be computed using (2).

Another example of a watermarking scheme is proposed in [88], where the feature extraction function $Ext(., K_{ext})$ consists of the scaling parameters of the Batch Normalization (BN) weights $W_\gamma = (\gamma^1, \gamma^2, ..., \gamma^l)$ (defined in Equation (7)) with l channels, which are chosen according to the secret position parameter K_{ext}.

$$y^i = \gamma^i \times x^i + \beta^i \tag{7}$$

where γ^i and β^i are the scaling and bias parameters in channel i for the BN layer, respectively, and x^i is the input of the BN layer. The projection function $Proj(., K_{Proj})$ in [88] is designed to insert a watermark $b \in \{0,1\}^l$, where l is the length of the watermark, and it is defined as follows:

$$Proj(W_\gamma, K_{Proj}) = Sgn(W_\gamma K_{Proj}) \in \{0,1\}^l \tag{8}$$

where K_{Proj} is smiliar to the scheme of Uchida et al. [15], which represents a secret random matrix of size $(|w|, l)$, and $Sgn(.)$ is the sign function:

$$Sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{Otherwise.} \end{cases} \tag{9}$$

The Hinge loss is used as the distance measure d for the watermarking regularization E_{wat}, which is given by:

$$d(b, y) = -\sum_{j=1}^{l} \max(\mu - b_i y_i, 0) \tag{10}$$

where μ represents the parameter of the Hinge loss as defined in [89]. Similar to the scheme proposed by Uchida et al. in [15], and utilizing the information about d, $Proj(., K_{Proj})$, and $Ext(., K_{ext})$, the watermarking loss E_{wat} for the purpose of watermarking a target model M can be computed using Equation (2).

On the other hand, the Black-Box setting assumes that the owner can perform the verification process only through an API: he can interact with the model only by giving inputs and receiving associated predictions. It also knows that the owner watermarks the model by changing its behavior. The common technique consists of training the model using a trigger set $T = (X_i, Y_i)_{i=1}$, which is composed of crafted inputs X_i with their associated outputs Y_i [75]. During each epoch, instead of giving a batch only from the train set to the model, we give a concatenation between a batch from the train set and the trigger set.

For example, Zhang et al. [74] propose using the same technique but with different types of inputs. They try three different types of images for the trigger set:

1. Content Watermarking: Incorporating meaningful content into images from the training dataset. The model should be activated by this content and provide the corresponding predefined label. In our example, the text "TEST" serves as the trigger for the model.
2. Unrelated Watermarking: Images that are irrelevant from the main task of the model. Each image has an associated label (like in [75]) or each sample can have its specific output. In our example, some images from the MNIST dataset are used to trigger the model.
3. Noise Watermarking: Adding a specific noise to images from the train set. Then, the model classifies any images with this specific noise as a predefined label. In our example, we add a small Gaussian noise to trigger the model.

The trigger set can also be built using adversarial examples. The authors of [77] proposed using a trigger set composed of two adversarial examples :

1. True adversaries: Samples that are misclassified by the model while being close to being well classified.
2. False adversaries: Well-classified samples from which we add an adversarial perturbation without changing their classification.

Figure 1 shows some examples of images from such trigger sets.

(A) **(B)** **(C)** **(D)** **(E)**

Figure 1. Illustration of trigger set samples derived from an original image. (**A**) original image; (**B**) content-based trigger sample; (**C**) unrelated-based trigger sample; (**D**) noise-based trigger sample; (**E**) adversarial-based trigger sample. All images are taken from the CIFAR10 dataset [90] except (**C**), which comes from the MNIST dataset [91].

Then, the model is trained to well classify the true adversaries with their true associated labels and the false adversaries with their labels. In Figure 1, we use the Fast Gradient Sign Method [92] to generate a possible false adversary.

To evaluate the performance of Black-Box watermarking embedding, we assess the accuracy of the model's output on the trigger set T and their labels as follows:

$$acc = \frac{1}{|T|} \sum_{(x,y)\in T} \mathbb{1}_{M^{wat}(x)=y} \tag{11}$$

3. Watermarking for Federated Learning

In this section, we introduce and define what is watermarking for federated learning including the different possible scenarios. Then, we formulate requirements for watermarking deployed in a federated learning context. Finally, we analyze the nine state-of-the-art methods.

3.1. Definition

In the context of centralized DNN watermarking, the goal is to simply prove the model's ownership subsequent to its training and release. In FL, ownership rights protection becomes a more complex problem due to the presence of multiple participants and multiple exchanges between them that have to be taken into account during the threat model formulation. To illustrate this issue, the authors in [17] show that the existing methods can naively be applied to FL in two manners: The first one consists of watermarking the model after the training is completed: for example, by using fine tuning to embed the watermark into the trained model. Without taking the fidelity requirement into account, any participant who receives the model (client or server) can steal the DNN before the last round. The second way is to embed the watermark before starting the training process. Even if the watermark will resist during the first rounds, it will be removed after several aggregation rounds. Thus, it is important to design a specific watermarking technique for FL that will be persistent from the first round until the model deployment.

We define watermarking for federated learning as the process for a participant or multiple participants to insert a watermark into the shared model. Following the client–server FL framework, the first question is to determine which part of the federation can watermark the model. Is the server more trustworthy since it manages the federation? Or are the clients more trustworthy, since their data are used? During this study, we distinguish three watermarking scenarios in the FL context, which we illustrate in Figure 2 according to who is watermarking the model.

($\mathbf{S_1}$) **Server:** The server is in charge of watermarking the global model.

($\mathbf{S_2}$) **Clients**: One or multiple clients watermark their updates in order to watermark the global model.

($\mathbf{S_3}$) **Server and clients**: The server and the clients collaborate to watermark the global model together.

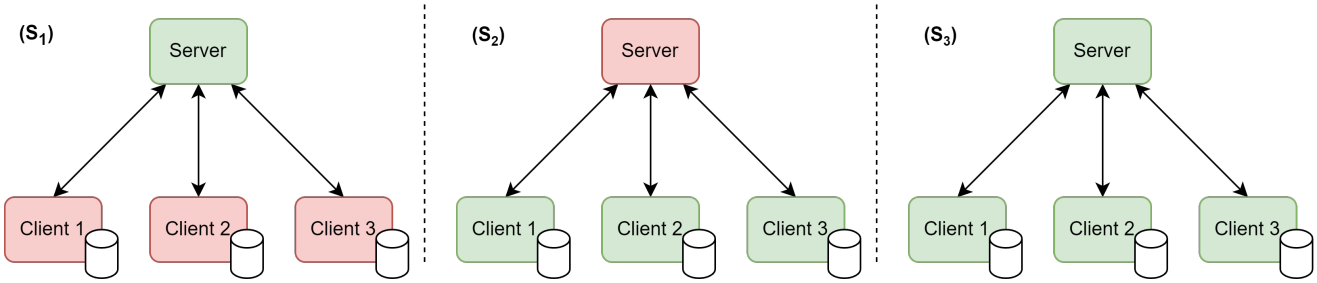

Figure 2. An example of the three possible scenarios of watermarking ($\mathbf{S_1}$–$\mathbf{S_3}$) in FL with one server and three clients. Green rectangles are the participants who follow the same watermarking procedure together. Red rectangles are those who are not enrolled in the watermarking procedure.

All watermarking requirements defined in Table 2 are also true in the federated context. However, due to the new constraints and the extension to several participants, we can add precision to five of them:

1. **Capacity:** When multiple clients want to insert their own message b_{C_i}, the watermarking technique needs to avoid possible conflict between the different inserted b_{C_i}. The number of bits needs to be then enough.
2. **Generality:** In a real FL scenario, many additional mechanisms are added for security and privacy such as robust aggregation functions (Section 4.2) or differential privacy [93] (Section 4.5). The watermarking technique must be applied independent of these mechanisms.
3. **Efficiency:** The cost generated by the embedding process is more crucial in FL. For example, in a cross-device architecture, clients have low computation power and they cannot perform many operations. The watermarking techniques must take this parameter into account.
4. **Secrecy:** If all parties are not enrolled in the watermarking process, the watermark should not be detected. In particular, if one or some clients are trying to watermark the global model, their updates need to be similar to benign updates. Otherwise, the server can use a defensive aggregation function to cancel the FL watermarking process (as described in Section 2.1).
5. **Robustness:** Since the model can be redistributed for any clients or the server, the watermark must track who is the traitor. Traitor tracing is the fact that each actor of the federation has a unique watermark that can be used to uniquely identify the owner in addition to a global watermark.

3.2. Related Works

In this section, we describe all state-of-the-art solutions of FL watermarking for IP protection. Note that all of the following papers are watermarking algorithms except for the last two, which focus on the ownership verification protocol. Table 3 gives a clear overview of existing methods depending on the previously described characteristics.

Table 3. FL watermarking taxonomy, categorized according to the access required to the verifier (White-Box or Black-Box), the entity that embeds the watermark (server, one or multiple clients), their compatibility with the cryptographic security tools, clients selection and poisoning defense mechanisms. ●: Compatible and tested, ◐: Compatible but not tested, -: Not compatible.

Existing Works	Verification	Watermarks Embedding	Security Tools Compatibility				Clients Selection	Poisoning Defense	
			DP	HE	MPC	TEE		Aggregation	Anomaly Detector
WAFFLE [17]	Black-Box	Server	◐	-	◐	◐	◐	◐	◐
FedIPR [18]	White-Box and Black-Box	Client(s)	●	◐	◐	◐	●	●	◐
FedTracker [94]	White-Box and Black-Box	Server	◐	-	◐	◐	◐	◐	◐
Liu et al. [95]	Black-Box	Client	◐	●	◐	◐	◐	-	-
FedCIP [96]	White-Box	Client(s)	◐	◐	◐	◐	●	◐	◐
FedRight [97]	White-Box	Server	◐	-	◐	◐	◐	◐	◐
Yang et al. [98]	Black-Box	Client	◐	●	◐	◐	◐	-	-
FedZKP [96]	White-Box	Client(s)	◐	◐	◐	◐	◐	◐	◐
Merkle-Sign [99]	White-Box and Black-Box	Server	◐	-	◐	◐	◐	◐	◐

3.2.1. WAFFLE

WAFFLE [17] is the first DNN watermarking technique for FL. The security hypothesis presented in the paper assumes that the server is a trusted party that embeds the watermark ($\mathbf{S_1}$) using a Black-Box watermarking technique using a trigger set. Clients cannot backdoor, poison, or embed their own watermarks since they are incentivized to maximize global accuracy. The adversary can only save the model and apply the post-processing technique as described in Section 2. Any trigger set that does not need any knowledge of the client's data can be used, but the authors present a specific set that is more suitable for FL: the WAFFLEPattern. WAFFLEPattern is defined as a set of images containing random patterns with a noisy background. Basically, the server will embed the watermark into the global model using the two following functions:

- **PreTrain**: Train an initialized model with the trigger set until the model has good accuracy on this set.
- **ReTrain**: Fine-tune the model with the trigger set until the model has learned the watermark.

PreTrain is used to embed the watermark in the model before the first round. During each round, after the aggregation process, the server uses **ReTrain** to re-embed the watermark into the model. The authors evaluated the robustness of their approach against various common post-processing techniques like fine-tuning and pruning and different types of attacks such as evasion attacks and backdoor detection using Neural Cleanse [100]. Figure 3 presents the learning curve of our WAFFLE implementation with 10 clients training a VGG16 model [101] on CIFAR10 [90]. We employed an IID configuration in which each client had an equal amount of training data with the same distribution. The server had access to the testing set, which exhibited a distribution similar to the training set. This accessibility enabled the server to evaluate the model at each round. As depicted in Figure 3, after 200 rounds, the watermarked model achieved a convergence with an accuracy of 0.84 on the testing set and a watermark detection rate close to 1.00 both before and after the **ReTrain** global model update by the server.

3.2.2. FedIPR

FedIPR [18] is both a Black-Box and White-Box technique. This technique allows all clients to embed their own watermark in the global model ($\mathbf{S_2}$) without sharing secret information. In terms of security hypothesis, the server is assumed to be honest but curious, while clients can be considered as malicious due to the fact they can interfere in the FL process. Each watermark can be described as follows:

- **Black-Box Watermark**: Each client generates a trigger set using the Projected Gradient Descent technique in a small Convolutional Neural Network (CNN) trained with the local data.

- **White-Box Watermark**: Each client generates a random secret matrix and a location in the Batch-Normalization layers to embed its message.

Both White-Box and Black-Box watermarks are inserted using an additional loss during the local training. For **Black-Box Watermark**, the loss is exactly the same as the loss used for the main task but with a batch of the trigger set as input. For the **White-Box Watermark**, the loss used is a Hinge-like loss [102] between the original message and the reconstructed message. They evaluate their solution in terms of robustness and additionally cover ambiguity and overwriting attacks. This scheme is the only one that has been tested under a federation with "free-riders", i.e., clients that typically send random model weights (e.g., Gaussian noise).

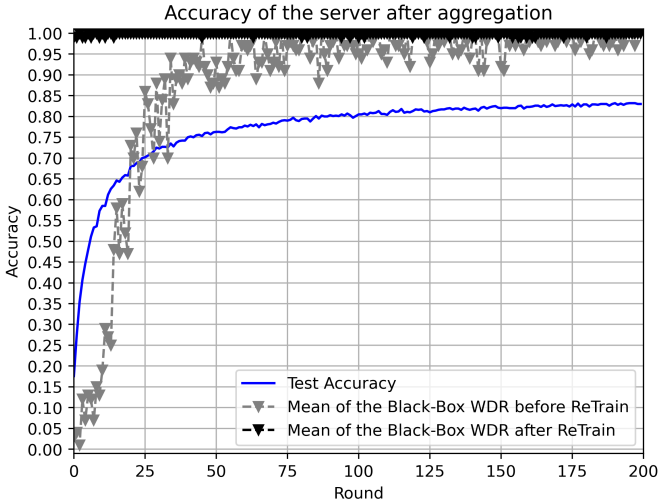

Figure 3. Progression of test accuracy and watermark detection rate during the training of a VGG16 [101] model with WAFFLE. The watermark detection rate is shown both before and after the **ReTrain** operation, as calculated using Equation (11).

3.2.3. FedTracker

FedTracker [94] is a watermarking technique that allows the server to embed a global Black-Box watermark ($\mathbf{S_1}$) but also a White-Box watermark specific to each client. The server is assumed to be a trusted party that has no access to natural data related to the primitive task. Some clients can be malicious and others are assumed to be honest. Malicious clients can copy and distribute the model, but they follow the training process to maximize global model accuracy. Each watermark can be described as follows:

- **Global Black-Box Watermark**: A trigger set is generated using the WAFFLEPattern method [17].
- **White-Box Watermark**: The server generates a random secret matrix and a fingerprint for each client.

After the aggregation, the server embeds the **global Black-Box watermark** using the intuition of Continual Learning [103] to avoid forgetting the main task. Then, for the **White-Box Watermark**, the loss used is a Hinge-like loss between the original message b and the reconstructed message $\tilde{b}$. Authors demonstrate robustness against fine-tuning, pruning, quantization, and overwriting.

3.2.4. Scheme of Liu et al.

Liu et al. [95] propose a client-side Black-Box watermarking scheme ($\mathbf{S_2}$). This technique is designed to embed a watermark only from one client (which represents the initiator of the FL). This framework is designed to be used under HE, since the server is not trusted. The authors assume also that other clients can be malicious. During the verification process, a fully trustworthy third party is recruited to be the arbitrator. The initiator creates a trigger

set composed of Gaussian noise images with a given label as a trigger set. Then, the client's model will overfit with this set like in [75]. To tackle the fact that this particular client will probably not be selected at each round, the authors introduce a scaling factor

$$\lambda = \frac{N}{n} \tag{12}$$

where N is the number of clients and n is the number of clients selected at each round. The client will then send its model weights multiplied by λ. According to the authors, this will be approximately equivalent to the case in which this client is selected every iteration and the watermark will be embedded more easily. The authors only tested fine-tuning, pruning, and quantization as possible attacks to defeat their watermarking solution.

3.2.5. FedRight

FedRight [97] is a solution for the server to fingerprint the model in the FL framework ($\mathbf{S_2}$). DNN fingerprinting is a process in which instead of embedding a watermark in the model, we extract a fingerprint to identify this model [104]. The security assumptions presented by the authors assume that the server is trusted and honest while clients may not be trustworthy. To do so, the server generates adversarial examples from a set of inputs (key samples). Then, the server extracts the probability distribution of each prediction and feeds it to a detector with the key sample target. Then, during the verification process, this detector is used to predict whether the corresponding model is a good one or not.

3.2.6. FedCIP

FedCIP [96] is a White-Box watermarking framework that is compatible with FL security aggregation and allows tracking traitors. This solution allows the clients to watermark the FL model ($\mathbf{S_2}$). They assume that the server is honest but curious, while model theft can be undertaken by any malicious client. The server cannot directly modify the model's parameters. The last security hypothesis relies on the fact that the trigger set should not rely on the original data. This technique is based on the concept of a unique watermark per client for a given number of rounds called a cycle. During a cycle, a portion cK of clients is designated to contribute during the following rounds. For each client, the watermark is unique during the cycle. If the round is the first iteration of the cycle, the algorithm will quickly replace the previous watermark with the new one. Otherwise, a small update is made to reinforce the watermark. The server divides the model parameters for all clients to avoid watermark conflicts, and it records the participants at each round, which is used for the verification process.

The verification is then performed using the "federated watermark", which is a concatenation of the watermark of all the contributors for a given cycle. If a client leaks the model multiple times during the FL, the authors point out that they can find the traitor by computing the intersection of the participants that have a high watermark detection rate with the leak models. In particular, they can identify the traitor if $c^n K \leq 1$ where n is the number of traitor leaks.

3.2.7. Yang et al. Scheme

Yang et al. [98] proposed a Black-Box watermarking framework based on Liu et al. [95] solution (described in Section 3.2.4) ($\mathbf{S_2}$). This approach has been developed to allow the client initiator of the federated learning (FL) process to embed a watermark for all the federations. More clearly, other clients and the server are not authorized to protect the model. Additionally, this method is compliant with the framework where homomorphic encryption (HE) is used due to a lack of trust in the server. The main contribution of this paper is the proposed trigger set schema. Instead of using a classical Gaussian noise-based trigger set, they build images using permutation-based secret keys and noise-based patterns. The authors claim and demonstrate that this trigger set is resistant to ambiguity attacks.

Whether the adversary tries to brute force the secret key or generate a new trigger set with its own key, both ways are hard to accomplish.

3.2.8. Merkle-Sign

Merkle-Sign [99] is a framework focusing on ownership verification in a collaborative client–server setting ($\mathbf{S_3}$). The authors propose a public verification protocol that uses the Merkle-tree [105]. In this framework, the server uses at each round an embedding function to insert two pieces of identity information (i.e., keys) into M_G: one that identifies the server and the other one that identifies the client that will receive the model. In parallel, the server uploads the tuple of keys and the tuple of verification function (which are generated by the watermarking embedding function) into a Merkle-tree with the recording time. In the final round, the server embeds also all keys generated by the clients into M_G and updates the Merkle-tree. This framework is also compatible in a peer-to-peer context. The associated Black-Box watermarking scheme relies on the training of an auto-encoder [106] (AE) for each client. Then, the server averages the received AE to obtain a final AE from which it can generate a trigger set using the keys as input for the decoder part.

In the Merkle-sign scheme, the server is trusted, and clients are considered honest but curious. The authors explore various attacks to assess the verification protocol's robustness, including ambiguity, spoiling, and traitor-tracing attacks. Ambiguity attacks can be countered by using an authorized time server or decentralized consensus protocol for timestamp authorization [67]. Security against spoil attacks involves applying multiple watermarks to the model, allowing clients to prove ownership even if an adversary spoils one watermark. For traitor tracing, the server embeds unique watermarks for each client, enabling successful ownership declaration over pirated models. However, the paper lacks a protocol for identifying traitor clients after training has terminated, as the model is watermarked using information from all clients simultaneously.

3.2.9. FedZKP

The authors in [88] propose a verification protocol based on the zero-knowledge proof and specifically on xLPN ZKP [107] called FedZKP. FedZKP does not require a trusted verifier to protect the confidentiality of the credentials, which reduces the risk of credential leakage. The security hypothesis presented in the paper assumes that clients are trusted entities, while the server and verifying authority are considered honest but curious ($\mathbf{S_2}$). Watermarking is conducted by the clients, each equipped with a public and private key. The hash of all clients' public keys is then embedded into the model using a White-Box mode. The paper examines four attacks: fine-tuning, pruning, and targeted destruction attacks. The experimental results demonstrate the watermark's robustness against these attacks. The last attack discussed in the paper involves a scenario where a user is required to prove ownership of a DNN model by sharing their public key with the trusted authority. However, this opens up the possibility for an adversary who intercepts the key to falsely claim ownership of the model. To address this vulnerability, the verification authority implements a zero-knowledge proof protocol, ensuring that only the individual possessing the private key associated with the public key in the watermarking can successfully demonstrate ownership.

It is worth noting that while the majority of the proposed solutions focus on IP protection, some papers explore alternative approaches using FL watermarking. For instance, WMDefense [108] utilizes FL watermarking to detect malicious client updates by assessing the degree of watermark recession. In this section, we have covered solutions encompassing both Black-Box/White-Box and client(s)/server watermarking with various ownership verification schemes. The summary of these solutions is presented in Table 3. Each solution is categorized based on the verification process (White-Box/Black-Box), the entity performing the watermarking (client(s) or server), compatibility with standard security tools as well as the client selection, and compatibility with poisoning defense mechanisms. After reviewing Table 3, it becomes apparent that most solutions have not undergone testing with security

tools such as differential privacy (DP) or homomorphic encryption (HE). Furthermore, none of the solutions have been subjected to testing involving multi-party computation (MPC) and trusted execution environments (TEEs). This same observation applies to poisoning defense mechanisms, as comprehensive testing across various solutions in this regard is yet to be explored.

4. Discussion

In this section, we identify and discuss specific challenges related to watermarking in FL. In particular, we evaluate how existing methods deal or not with these new challenges.

4.1. Trigger-Set-Based Watermarking on the Server Side

Black-Box watermarking consists of changing the behavior of the model. To do so, most methods let the model overfit on the trigger set by adding a regularization term in the loss function. Usual DNN watermarking techniques can easily be applied in $\mathbf{S_2}$ and $\mathbf{S_3}$. However, it is not so easy in $\mathbf{S_1}$. When the client C_k wants to watermark its model M_{C_k}, he can rely on two things:

1. Have access to its private dataset D_{C_k};
2. Train the model on both the main task dataset D_{C_k} and its trigger set T_{C_k} at the same time.

A large number of Black-Box watermarking techniques need to build T_{C_k} using $D_{C_k} = (X_i, Y_i)_{i=1}$ (as discussed in Section 2.2). Using such techniques is motivated by the fact that training the model from these datasets is multi-task learning. Building T_{C_k} from D_{C_k} helps to reduce the negative impact on learning from two domains. Moreover, learning these two tasks together avoids catastrophic forgetting [109].

In $\mathbf{S_1}$, since the server does not have its own dataset, it cannot perform such types of watermarking. The choice is limited by using unrelated or noise-based inputs such as WAFFLEPattern [17]. He can also generate them from auto-encoders provided by clients such as in Merkle-Sign [99]. However, as we said before, learning two tasks separately can be negative for Fidelity. FedTracker [94] is the only server-side framework in which they use Continual Learning [103] and a global gradient memory to not interfere with previously learned tasks. Table 4 exhibits the type of trigger set used by each Black-Box technique with their resilience against two attacks.

Table 4. Black-Box FL watermarking techniques with their trigger set type and their resilience against Black-Box attacks. ●: Resilient, -: Not tested.

Existing Works	Watermarks Embedding	Trigger Set	Evasion Attack	Backdoor Detection [100]
WAFFLE [17]	Server	Noise	●	●
Merkle-Sign [105]	Server	Generated by Auto-Encoder	-	-
FedTracker [94]	Server	Noise	●	●
FedIPR [18]	Client(s)	Adversarial Examples	-	-
Liu et al. [95]	Client	Noise	-	-
Yang et al. [98]	Client	Noise	-	-

In addition, this limitation increases vulnerability to evasion attacks. More clearly, during the Black-Box verification process, the owner will ask about the suspicious Application Programming Interface (API) that possibly contains his DNN. The owner will send samples from the trigger set and check if the corresponding labels are the same as the ones learned by its model. However, the attacker may evade this verification using a query detector: such a type of detector analyzes the input of the request to check if this latter is consistent with the task of the model. If it is not consistent, the query is rejected [110]. Since the trigger set is built using images that are qualified to be Out-Of-Distribution (O.O.D), this implies an easier detection for the attacker who stole the model [111]. WAFFLE's authors show that their method is resilient against evasion attacks with the WAFFLEPattern trigger set

(see Section 3.2.1), but it is also under some constraints especially in terms of the number of clients who share their data to build the detector.

4.2. Aggregation Functions

The most common aggregation function is `FedAvg` [7], which consists of averaging clients' parameters after they perform multiple epochs on mini-batches. Each client weight matrix is multiplied by a scaling factor defined as $\frac{n_{C_k}}{n}$ where n_{C_k} is the number of samples in D_{C_k} and $n = \sum_k^K n_{C_k}$. Many aggregation functions emerged to meet various challenges in FL. Since the clients do not necessarily know which aggregation function the server is using, the proposed methods must be independent of this parameter.

For the Byzantine-attacks problem in which one or multiple clients try to disturb the FL process, these attacks can be simple noise weights or complex label-flipping backdoors. To leverage this problem, multiple aggregation functions appear to select only benign updates such as `Krum` [23] `Trim-mean` or `Bulyan` [24]. Since clients' watermarking techniques are sensitive to embedding a watermark b and the trigger set T, they keep their updates far from each other. A part of the updates can be rejected for this reason if we use defensive aggregation techniques. As an example, FedIPR shows that the **White-Box watermark** results are similar to `FedAvg` with a detection rate of 97.5% using `Trim-mean`. However, the **Black-Box watermark** reaches only 63.25% of the watermark detection rate at the end of the FL process. Even if this score is enough to detect plagiarism, using a defensive aggregation function has a huge impact on the watermark. The study of Liu et al. [95] and its extension by Yang et al. [98] have not tested yet their solution with a defensive aggregation function because they use HE. But we can guess that multiplying weights by a big scaling factor λ can be easy to detect for `Krum` as a Byzantine attack, as shown in a similar example [37].

Another problem is that `FedAvg` performs well when the data are statistically homogeneously distributed among the clients. However, in real use cases, data are heterogeneous, which may lead to difficulty for the model to converge using `FedAvg` [20]. As we can see from Table 5, existing watermarking techniques for FL have not evaluated aggregation methods that tackle this problem such as `FedProx` [20], `FedNova` [21] or `SCAFFOLD` [22]. If we want to use the proposed solution in a real secure FL framework, these methods need to be tested in a such context, which is actually not the case.

Table 5. Applications and parameters considered for experimenting with each FL watermarking method. ●: Tested, -: Unmentioned.

Existing Works	Framework	Task	Maximum Number of Clients	Aggregation Function	Non-I.I.D Repartition of Training Set
WAFFLE [17]	PySyft, Pytorch	Image Classification	100	`FedAvg`	●
FedIPR [18]	Pytorch	Image Classification, NLP	100	`FedAvg, Trim-Mean, Bulyan, Multi-Krum`	●
FedTracker [94]	-	Image Classification	50	`FedAvg`	●
Liu et al. [95]	-	Image Classification	100	`FedAvg`	-
FedCIP [96]	-	Image Classification	10	`FedAvg`	●
FedRight [97]	-	Image Classification	100	`FedAvg`	-
Yang et al. [98]	-	Image Classification	100	`FedAvg`	-
FedZKP [96]	-	Image Classification	50	`FedAvg`	-
Merkle-Sign [99]	-	Image Classification	200	`Gradient Average` [105]	-

4.3. Clients Selection

For the sake of communication efficiency, especially when dealing with a large number of clients, a strategy involves selecting a fixed number of clients during each round, as outlined in Section 2.1. This process entails the random selection of cK clients, where c lies in the interval $(0, 1)$. It is important to note that this client selection mechanism can significantly impact watermarking. In the context of ($\mathbf{S_1}$), the impact of this mechanism is expected to be minimal. This is because in ($\mathbf{S_1}$), the server embeds the watermark during each round regardless of the value of cK. On the contrary, ($\mathbf{S_2}$) is more susceptible to the influence of the client selection process. Given that every client aims to insert its individual watermark, clients not selected run the risk of having their watermarks compromised or

eliminated in the global model. However, this effect has not been extensively explored in most research papers.

Researchers behind the FedIPR [18] study have revealed that with $c > 0.2$, both White-Box and Black-Box watermark detection rates approach 100%. In cases where $c \leq 0.2$, the detection rate for a feature-based watermark remains close to 100%, whereas the detection rate for the backdoor watermark drops to 62%. In contrast, the FedCIP [96] framework is intentionally crafted to incorporate a client selection routine. As demonstrated, this framework showcases strong performances across various watermarking requirements.

4.4. Cross-Device Setting

All proposed papers are treating the case in which we have a small number of clients. According to Table 5, the worst scenario is tested in Merkle-Sign in which 200 clients are enrolled in the federation. However, there is no solution that takes into account the cross-device setting. In this setting, a large number of clients (up to 10^{10} devices) are enrolled in the FL procedure [19]. These clients are not always reachable, and they have a low dedicated computational power, which is defined as a performance condition by the authors of WAFFLE.

In the Black-Box setting, the problem can come from the low computation power that does not allow the clients to perform more computations to increase the batch size using trigger-set methods. For the White-Box setting, the bottleneck would be the **Capacity**, as mentioned in Section 3.1, in particular in cases where the proposed methods are tested using Normalization layers such as FedIPR or FedTracker, which limits the overall embedding capacity. It makes it difficult for each client to embed its watermark b without conflicting with the face of other clients' watermarks.

4.5. Differential Privacy and Homomorphic Encryption

Since federated learning (FL) ensures the privacy of clients' data by sharing the model or gradient updates between the server and clients (or directly among clients), there are concerns about potential attacks, such as membership inference, which can reveal the presence of specific data points [112]. To address this issue, differential privacy (DP) is often employed, providing robust privacy guarantees for algorithms working on aggregate databases [93,113]. In FL, a common DP technique involves introducing Gaussian random noise to the gradients sent to the aggregator, adding an extra layer of privacy protection.

Another cryptographic measure to enhance security in client–server FL is homomorphic encryption (HE) [114–116]. In this approach, clients can protect their model updates using their public keys, encrypting the data before sending it to the server. The server performs the model aggregation (usually using `FedAvg`) in the encrypted space, ensuring that the server cannot misuse the privilege of having access to individual client updates to learn about private datasets. Clients can then use their private keys to decrypt the aggregated global model once it is received.

The combination of differential privacy and homomorphic encryption addresses privacy and security concerns in different aspects of FL. While DP protects against attacks attempting to extract sensitive information from gradient updates, HE safeguards the clients' data during aggregation, preventing unauthorized access by the server. In the context of watermarking, the use of HE is particularly suitable for scenario ($\mathbf{S_2}$), as clients can access their model parameters to embed watermarks securely. However, a challenge remains for scenario ($\mathbf{S_1}$), as the server cannot decrypt the model parameters to perform watermarking embedding.

4.6. Watermarking for Non-Client–Server Framework

Decentralized FL ($\mathbf{S_2}$) is an interesting framework in which clients do not need a server to perform the model aggregation. The proposed methods seem to be applicable to this setting, since watermarking the model from the client side does not require the server. However, Merkle-Sign is a unique solution that extends to the decentralized setting. We

can also cite split learning, in which the server has a part of the network and clients have another part. Performing White-Box watermarking as in [15] can be more difficult for the server or for clients. In both cases, they have access to a part of the model parameters that can be arbitrarily small.

In the U-shape split learning architecture, the server has only the middle part of the model and the clients have the first and last layers. In this setting, performing a Black-Box watermarking on the server side is hard, since it cannot use its inputs and labels on the model.

4.7. Attacks from Clients and/or Server Sides

When we analyze ($\mathbf{S_1}$) and ($\mathbf{S_2}$) scenarios, we can see that each one has different parameters to play with whether for watermarking or disrupting it. The server can control the selected participants or how to aggregate the model parameters. It has also sometimes a clear view of clients' parameters at each round if no safety measures are considered in the learning process. However, it does not have data, and it cannot fully control whether clients are strictly following the training process. On the other hand, clients have their private dataset, and they can send the update that they want. Nevertheless, they have no control over what happens with their updates at the server level.

If the server wants to avoid a subset of clients to watermark the model, it can use methods proposed for Byzantine attacks [32] detection. In particular, attacks that consist of multiplying the weights by a scaling factor to replace or have a bigger impact in the global model are easy to detect [37]. The proposed method by Yang et al. [95], without HE, is then easily removable, and the global model will not be watermarked. A solution to catch backdoor models was presented in [117,118]. Then, all proposed solutions that rely on a backdoor-based watermarking can be rejected.

Clients can also try to disturb the watermarking process. FedIPR ($\mathbf{S_2}$) [18] authors present the free-riders problem in which some clients do not contribute to the training of M_G and the watermarking process. Even if with their solution they have no important impact on the watermarking, no testing has yet been performed on ($\mathbf{S_1}$). Another attack that is specific to FL as described in FedRight [97] and WAFFLE [17], consists of the fact that multiple clients will use their models and private datasets. As mentioned in Section 4.1, an evasion attack works better when multiple datasets are used to train the detector. But it is also possible to fine-tune the model with the combined dataset.

To sum up, the key findings of our analysis are the following:

- Client-side solutions—None of these solutions have undergone testing with poisoning detection mechanisms such as anomaly detection, and only a few of them consider defensive aggregation functions like `Krum` and `GeoMed`.
- Server-side solutions—Black-Box watermarking is still challenging since the training data are not available for the server. And no watermarking algorithm has been found compatible with cryptographic tools like homomorphic encryption (HE).
- Other FL architectures—Only one method is compatible with peer-to-peer FL, and no solution is proposed for split learning. No solution has been tested with the cross-device setting, despite its importance in FL.
- Data distribution—Only a small portion of the proposed solutions have been tested in non-I.I.D scenarios, and none have been experimented with in vertical data distribution.
- FL framework—Despite the number of proposed FL watermarking solutions, no FL framework integrates such a tool as presented in Table 1.

5. Perspectives

Based on existing methods described in Section 3 and challenges related to FL watermarking in the previous section, one can list the following future research directions to investigate:

- As mentioned by Boenisch [52], like most centralized watermarking techniques, FL watermarking is designed and tested for image classification tasks. FedIPR [18] is the only method that extends the experimental testing to Natural Language Processing (NLP) tasks. Thus, it is of interest to design new FL watermarking techniques for other tasks such as Object Detection [119], Semantic Segmentation [120], Regression [121], Conversational [122], and so on.
- It is worth noting that each state-of-the-art method has been experimented with non-standardized parameters, as illustrated in Tables 3, 5, and 6. Furthermore, the majority of authors have not shared their source codes, making reproducibility and method comparison challenging. Table 6 summarizes the performance of each method. Given that they were tested with different datasets, model architectures, and parameters, making a fair comparison is inherently difficult. Nonetheless, it is evident that these methods have achieved commendable performance. As a final observation, it is worth mentioning that the tested models and datasets may not fully align with the realities of federated learning frameworks in domains such as health care [123] and IoT [124], as discussed in Section 4.4.
- Merkle-Sign [105] and FedZKP [88] are the only two proposals that present a rigorous ownership verification process based on cryptographic security. Such a verification protocol is also a research axis that needs to be investigated. This is of interest, for example, to counteract ambiguity and spoiling attacks or to enable traitor tracing.
- In Table 1, we have listed eleven open-source federated learning frameworks. Half of these frameworks are designed for research purposes, while the other half serve industrial purposes. Unfortunately, none of these solutions includes a watermarking feature or a pipeline to facilitate the integration and experimentation of such protection measures, as has been completed for differential privacy, homomorphic encryption, multi-party computation, and trusted execution environments [8,11–13,29,30]. Furthermore, as discussed in Section 4.5, it is worth noting that proposed federated learning watermarking solutions do not always account for the presence of these other security mechanisms within their frameworks.

Table 6. Experimental results for each method based on authors' parameters. Due to variations in these parameters and the use of different architectures, conducting a fair comparison is challenging. In the Baseline, Fine-Tuning, and Pruning columns, the first number represents the accuracy on the test set (Acc.), while the value in parentheses indicates the watermark detection rate (WDR). -: Not mentioned.

Existing Works	Model Used	Dataset	Baseline	Fine Tuning	Pruning
			Acc. (WDR)	Acc. (WDR)	Acc. (WDR)
WAFFLE [17]	VGG	CIFAR10	85.8 (99.0)	85.6 (96.2)	85.0 (47.0)
		MNIST	98.9 (100.0)	98.7 (98.0)	97.0 (47.0)
FedIPR [18]	AlexNet	CIFAR10	91.7 (99.0)	91.7 (98.0)	78.0 (99.0)
	ResNet	CIFAR100	76.9 (99.0)	- (-)	- (-)
FedTracker [94]	VGG	CIFAR10	88.6 (99.1)	- (99.1)	83.0 (84.5)
	AlexNet	CIFAR10	84.3 (82.6)	- (82.6)	73.0 (52.4)
Liu et al. [95]	VGG	CIFAR10	84.2 (100.0)	- (87.0)	78.0 (71.0)
	LeNet	MNIST	99.3 (100.0)	- (97.5)	98.4 (100.0)
FedCIP [96]	ResNet	CIFAR10	82.0 (100.0)	- (100.0)	- (92.0)
FedRight [97]	VGG	CIFAR10	86.0 (100.0)	- (87.0)	85.13 (62.85)
	LetNet	MNIST	96.0 (100.0)	- (100.0)	93.5 (58.1)
Yang et al. [98]	VGG	CIFAR10	82.4 (99.7)	- (99.7)	68.3 (87.9)
	LeNet	MNIST	99.2 (99.2)	- (99.2)	99.2 (100.0)
FedZKP [96]	AlexNet	CIFAR10	91.5 (100.0)	91.5 (99.0)	- (-)
	ResNet	CIFAR100	77.2 (100.0)	- (-)	51.0 (100.0)
Merkle-Sign [99]	ResNet	CIFAR10	89.1 (100.0)	- (-)	- (-)
		CIFAR100	62.5 (100.0)	- (-)	- (-)
		MNIST	99.6 (100.0)	- (-)	- (-)
		Fashion	93.8 (100.0)	- (-)	- (-)

6. Conclusions

In this paper, we have shown through the existing methods how DNN watermarking is an efficient solution for federated learning IP protection. We also point out that several problems have to be considered. For the first time, these techniques are used mainly for image classification, while emerging FL models address more complex tasks (e.g., semantic segmentation, object detection, text generation) while considering constraints associated with specific fields such as the health-care domain, IOTs, and so on. All these issues lead to difficulties in evaluating such methods in real-world FL scenarios. Second, these methods often take into account non-IID data in their experiments without using aggregation functions designed for such data distribution. That is in particular the case in the context of vertical distribution or split learning, which are two unexplored areas despite their interest in the FL field. Another important point is that the source code of the methods is rarely given. This makes it difficult to reproduce the experiments. If the community merges to find a standard for the implementation and integration into existing FL frameworks, development and interest in FL watermarking will be enhanced. Furthermore, one needs to anticipate that these solutions will be seamlessly integrated into secure FL frameworks, enabling their deployment across a wider range of use cases. However, the authors of FL watermarking methods rarely take into account FL privacy-preserving mechanisms. So, future works need to combine security tools such as homomorphic encryption, differential privacy, and multi-party computation to clearly identify possible interferences with FL watermarking. It is our hope that this paper will contribute to the advancement of research in the area of Intellectual Property (IP) protection. In future work, we will implement the presented methods to compare them in a unified framework. Showing the results of each technique using the same FL parameters will allow making a deeper comparison and highlight the most realistic techniques in a real-world FL scenario.

Author Contributions: Conceptualization, M.L., R.B., K.K., V.T., O.B. and G.C.; methodology, M.L., R.B. and G.C.; formal analysis, M.L. and R.B.; investigation, M.L. and R.B.; resources, M.L. and R.B.; writing—original draft preparation, M.L. and R.B.; writing—review and editing, M.L., R.B., K.K. and V.T.; visualization, M.L. and R.B.; project administration, O.B. and G.C.; funding acquisition, O.B. and G.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the European Union under Grant Agreement 101070222. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission (granting authority). Neither the European Union nor the granting authority can be held responsible for them. This work is also supported by the CYBAILE industrial chair, which is led by Inserm with the support of the Brittany Region Council.

Funded by the European Union

Data Availability Statement: The data used in this article are publicly available.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Singh, K.; Booma, P.; Eaganathan, U. E-commerce system for sale prediction using machine learning technique. *Proc. J. Physics Conf. Ser.* **2020**, *1712*, 012042. [CrossRef]
2. Conze, P.h.; Daho, M.E.H.; Li, Y.; Brahim, I.; Le Boité, H.; Massin, P.; Tadayoni, R.; Cochener, B.; Quellec, G.; Lamard, M.; et al. Time-aware deep models for predicting diabetic retinopathy progression. *Investig. Ophthalmol. Vis. Sci.* **2023**, *64*, 246–246.

3. Mallozzi, P.; Pelliccione, P.; Knauss, A.; Berger, C.; Mohammadiha, N. Autonomous vehicles: State of the art, future trends, and challenges. In *Automotive Systems and Software Engineering: State of The Art and Future Trends*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 347–367.
4. Regulation, P. General data protection regulation. *Intouch* **2018**, *25*, 1–5.
5. Piper, D. *Data Protection Laws of the World*; DLA Piper: London, UK, 2019.
6. Chen, J.; Sun, J. Understanding the chinese data security law. *Int. Cybersecur. Law Rev.* **2021**, *2*, 209–221. [CrossRef]
7. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, Melbourne, VIC, Australia, 19–20 August 2017; pp. 1273–1282.
8. Benaissa, A.; Retiat, B.; Cebere, B.; Belfedhal, A.E. TenSEAL: A library for encrypted tensor operations using homomorphic encryption. *arXiv* **2021**, arXiv:2104.03152.
9. Gehlhar, T.; Marx, F.; Schneider, T.; Suresh, A.; Wehrle, T.; Yalame, H. SAFEFL: MPC-friendly Framework for Private and Robust Federated Learning. *Cryptol. Eprint Arch.* **2023**, *2023*, 555.
10. Chen, H.; Laine, K.; Rindal, P. Fast private set intersection from homomorphic encryption. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1243–1255.
11. Dwork, C. Differential privacy: A survey of results. In Proceedings of the International Conference on Theory and Applications of Models of Computation, Xi'an, China, 25–29 April 2008; pp. 1–19.
12. Wei, K.; Li, J.; Ma, C.; Ding, M.; Chen, W.; Wu, J.; Tao, M.; Poor, H.V. Personalized Federated Learning with Differential Privacy and Convergence Guarantee. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 4488–4503. [CrossRef]
13. Zheng, W.; Cao, Y.; Tan, H. Secure sharing of industrial IoT data based on distributed trust management and trusted execution environments: A federated learning approach. *Neural Comput. Appl.* **2023**, *2023*, 1–11. [CrossRef]
14. Xu, Z.; Zhang, Y.; Andrew, G.; Choquette-Choo, C.A.; Kairouz, P.; McMahan, H.B.; Rosenstock, J.; Zhang, Y. Federated Learning of Gboard Language Models with Differential Privacy. *arXiv* **2023**, arXiv:2305.18465.
15. Uchida, Y.; Nagai, Y.; Sakazawa, S.; Satoh, S. Embedding watermarks into deep neural networks. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, Bucharest, Romania, 6–9 June 2017; pp. 269–277.
16. Yang, Q.; Huang, A.; Fan, L.; Chan, C.S.; Lim, J.H.; Ng, K.W.; Ong, D.S.; Li, B. Federated Learning with Privacy-preserving and Model IP-right-protection. *Mach. Intell. Res.* **2023**, *20*, 19–37. [CrossRef]
17. Tekgul, B.G.; Xia, Y.; Marchal, S.; Asokan, N. WAFFLE: Watermarking in federated learning. In Proceedings of the 2021 40th International Symposium on Reliable Distributed Systems (SRDS), Chicago, IL, USA, 20–23 September 2021; pp. 310–320.
18. Li, B.; Fan, L.; Gu, H.; Li, J.; Yang, Q. FedIPR: Ownership verification for federated deep neural network models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 4521–4536. [CrossRef] [PubMed]
19. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends® Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
20. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. *Proc. Mach. Learn. Syst.* **2020**, *2*, 429–450.
21. Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; Poor, H.V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7611–7623.
22. Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; Suresh, A.T. Scaffold: Stochastic controlled averaging for federated learning. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual Event, 13–18 July 2020; pp. 5132–5143.
23. Blanchard, P.; El Mhamdi, E.M.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1.
24. Guerraoui, R.; Rouault, S. The hidden vulnerability of distributed learning in byzantium. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 3521–3530.
25. Brendan McMahan, H.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. *arXiv* **2016**, arXiv:1602.
26. Roy, A.G.; Siddiqui, S.; Pölsterl, S.; Navab, N.; Wachinger, C. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv* **2019**, arXiv:1905.06731.
27. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv* **2018**, arXiv:1812.00564.
28. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated learning with non-iid data. *arXiv* **2018**, arXiv:1806.00582.
29. Mo, F.; Shamsabadi, A.S.; Katevas, K.; Demetriou, S.; Leontiadis, I.; Cavallaro, A.; Haddadi, H. Darknetz: Towards model privacy at the edge using trusted execution environments. In Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services, Florence, Italy, 18–22 May 2020; pp. 161–174.
30. Kanagavelu, R.; Li, Z.; Samsudin, J.; Yang, Y.; Yang, F.; Goh, R.S.M.; Cheah, M.; Wiwatphonthana, P.; Akkarajitsakul, K.; Wang, S. Two-phase multi-party computation enabled privacy-preserving federated learning. In Proceedings of the 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, VIC, Australia, 11–14 May 2020; pp. 410–419.

31. Fang, M.; Cao, X.; Jia, J.; Gong, N. Local model poisoning attacks to {Byzantine-Robust} federated learning. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Virtual Evenet, 12–14 August 2020; pp. 1605–1622.
32. Shi, J.; Wan, W.; Hu, S.; Lu, J.; Zhang, L.Y. Challenges and approaches for mitigating byzantine attacks in federated learning. In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 9–11 December 2022; pp. 139–146.
33. Huang, A. Dynamic backdoor attacks against federated learning. *arXiv* **2020**, arXiv:2011.07429.
34. Xie, C.; Huang, K.; Chen, P.Y.; Li, B. Dba: Distributed backdoor attacks against federated learning. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
35. Yin, D.; Chen, Y.; Kannan, R.; Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Proceedings of the International Conference on Machine Learning, PMLR, Atlanta, GA, USA, 16–21 June 2018; pp. 5650–5659.
36. Anass, E.M.; Gouenou, C.; Reda, B. Poisoning-Attack Detection Using an Auto-encoder for Deep Learning Models. In Proceedings of the International Conference on Digital Forensics and Cyber Crime, Boston, MA, USA, 16–18 November 2022; pp. 368–384.
37. Gu, Z.; Yang, Y. Detecting malicious model updates from federated learning on conditional variational autoencoder. In Proceedings of the 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Portland, OR, USA, 17–21 May 2021; pp. 671–680.
38. Li, S.; Cheng, Y.; Wang, W.; Liu, Y.; Chen, T. Learning to detect malicious clients for robust federated learning. *arXiv* **2020**, arXiv:2002.00211.
39. Cremonesi, F.; Vesin, M.; Cansiz, S.; Bouillard, Y.; Balelli, I.; Innocenti, L.; Silva, S.; Ayed, S.S.; Taiello, R.; Kameni, L.; et al. Fed-BioMed: Open, Transparent and Trusted Federated Learning for Real-world Healthcare Applications. *arXiv* **2023**, arXiv:2304.12012.
40. TensorFlow Federated: Machine Learning on Decentralized Data. Available online: https://www.tensorflow.org/ (accessed on 16 August 2023)
41. Ziller, A.; Trask, A.; Lopardo, A.; Szymkow, B.; Wagner, B.; Bluemke, E.; Nounahon, J.M.; Passerat-Palmbach, J.; Prakash, K.; Rose, N.; et al. Pysyft: A library for easy federated learning. *Fed. Learn. Syst. Towards-Next-Gener. AI* **2021**, *2021*, 111–139.
42. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A friendly federated learning research framework. *arXiv* **2020**, arXiv:2007.14390.
43. FATE: An Industrial Grade Federated Learning Framework. Available online: https://fate.fedai.org/ (accessed on 16 August 2023)
44. Reina, G.A.; Gruzdev, A.; Foley, P.; Perepelkina, O.; Sharma, M.; Davidyuk, I.; Trushkin, I.; Radionov, M.; Mokrov, A.; Agapov, D.; et al. OpenFL: An open-source framework for Federated Learning. *arXiv* **2021**, arXiv:2105.06413.
45. Ludwig, H.; Baracaldo, N.; Thomas, G.; Zhou, Y.; Anwar, A.; Rajamoni, S.; Ong, Y.; Radhakrishnan, J.; Verma, A.; Sinn, M.; et al. Ibm federated learning: An enterprise framework white paper v0. 1. *arXiv* **2020**, arXiv:2007.10987.
46. Roth, H.R.; Cheng, Y.; Wen, Y.; Yang, I.; Xu, Z.; Hsieh, Y.T.; Kersten, K.; Harouni, A.; Zhao, C.; Lu, K.; et al. Nvidia flare: Federated learning from simulation to real-world. *arXiv* **2022**, arXiv:2210.13291.
47. Federated Learning powered by NVIDIA Clara. Available online: https://www.nvidia.com/fr-fr/clara/ (accessed on 16 August 2023)
48. Xue, M.; Wang, J.; Liu, W. DNN intellectual property protection: Taxonomy, attacks and evaluations. In Proceedings of the 2021 on Great Lakes Symposium on VLSI, Virtual Conference and Exhibition, 22–25 June 2021; pp. 455–460.
49. Lukas, N.; Jiang, E.; Li, X.; Kerschbaum, F. SoK: How robust is image classification deep neural network watermarking? In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022; pp. 787–804.
50. Li, Y.; Wang, H.; Barni, M. A survey of deep neural network watermarking techniques. *Neurocomputing* **2021**, *461*, 171–193. [CrossRef]
51. Fkirin, A.; Attiya, G.; El-Sayed, A.; Shouman, M.A. Copyright protection of deep neural network models using digital watermarking: A comparative study. *Multimed. Tools Appl.* **2022**, *81*, 15961–15975. [CrossRef]
52. Boenisch, F. A systematic review on model watermarking for neural networks. *Front. Big Data* **2021**, *4*, 729663. [CrossRef]
53. Sun, Y.; Liu, T.; Hu, P.; Liao, Q.; Ji, S.; Yu, N.; Guo, D.; Liu, L. Deep Intellectual Property: A Survey. *arXiv* **2023**, arXiv:2304.14613.
54. Bouslimi, D.; Bellafqira, R.; Coatrieux, G. Data hiding in homomorphically encrypted medical images for verifying their reliability in both encrypted and spatial domains. In Proceedings of the 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Orlando, FL, USA, 16–20 August 2016; pp. 2496–2499.
55. Ernawan, F.; Ariatmanto, D. A recent survey on image watermarking using scaling factor techniques for copyright protection. *Multimed. Tools Appl.* **2023**, *2023*, 1–41. [CrossRef]
56. Pavan, A.; Somashekara, M. An Overview on Research Trends, Challenges, Applications and Future Direction in Digital Image Watermarking. *Int. Res. J. Adv. Sci. Hub* **2023**, *5*, 8–14.
57. Niyitegeka, D.; Coatrieux, G.; Bellafqira, R.; Genin, E.; Franco-Contreras, J. Dynamic watermarking-based integrity protection of homomorphically encrypted databases—Application to outsourced genetic data. In Proceedings of the International Workshop on Digital Watermarking, Jeju Island, Republic of Korea, 22–24 October 2018; pp. 151–166.
58. Hu, D.; Wang, Q.; Yan, S.; Liu, X.; Li, M.; Zheng, S. Reversible Database Watermarking Based on Order-preserving Encryption for Data Sharing. *ACM Trans. Database Syst.* **2023**, *48*, 1–25. [CrossRef]

59. Song, C.; Ristenpart, T.; Shmatikov, V. Machine learning models that remember too much. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 587–601.
60. Feng, L.; Zhang, X. Watermarking neural network with compensation mechanism. In Proceedings of the Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30 2020; pp. 363–375.
61. Li, Y.; Tondi, B.; Barni, M. Spread-Transform Dither Modulation Watermarking of Deep Neural Network. *arXiv* **2020**, arXiv:2012.14171.
62. Tartaglione, E.; Grangetto, M.; Cavagnino, D.; Botta, M. Delving in the loss landscape to embed robust watermarks into neural networks. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 1243–1250.
63. Chen, H.; Rouhani, B.D.; Fu, C.; Zhao, J.; Koushanfar, F. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In Proceedings of the 2019 on International Conference on Multimedia Retrieval, Ottawa, ON, Canada, 10–13 January 2019; pp. 105–113.
64. Wang, J.; Wu, H.; Zhang, X.; Yao, Y. Watermarking in deep neural networks via error back-propagation. *Electron. Imaging* **2020**, *2020*, 1–22. [CrossRef]
65. Rouhani, B.D.; Chen, H.; Koushanfar, F. Deepsigns: An end-to-end watermarking framework for protecting the ownership of deep neural networks. In Proceedings of the 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), ACM, Providence, RI, USA, 13–17 April 2019.
66. Fan, L.; Ng, K.W.; Chan, C.S. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1.
67. Li, F.; Wang, S. Secure watermark for deep neural networks with multi-task learning. *arXiv* **2021**, arXiv:2103.10021.
68. Bellafqira, R.; Coatrieux, G. DICTION: DynamIC robusT whIte bOx watermarkiNg scheme. *arXiv* **2022**, arXiv:2210.15745.
69. Kuribayashi, M.; Yasui, T.; Malik, A. White Box Watermarking for Convolution Layers in Fine-Tuning Model Using the Constant Weight Code. *J. Imaging* **2023**, *9*, 117. [CrossRef]
70. Lv, P.; Li, P.; Zhang, S.; Chen, K.; Liang, R.; Ma, H.; Zhao, Y.; Li, Y. A Robustness-Assured White-Box Watermark in Neural Networks. *IEEE Trans. Dependable Secur. Comput. (TDSC)* **2023**, 1–14. https://doi.org//10.1109/TDSC.2023.3242737. [CrossRef]
71. Rodriguez-Lois, E.; Perez-Gonzalez, F. Towards Traitor Tracing in Black-and-White-Box DNN Watermarking with Tardos-based Codes. *arXiv* **2023**, arXiv:2307.06695.
72. Chen, H.; Rouhani, B.D.; Koushanfar, F. BlackMarks: Blackbox Multibit Watermarking for Deep Neural Networks. *arXiv* **2019**, arXiv:1904.00344.
73. Vybornova, Y. Method for copyright protection of deep neural networks using digital watermarking. In Proceedings of the Fourteenth International Conference on Machine Vision (ICMV 2021), Rome, Italy, 8–12 November 2022; Volume 12084, pp. 297–304.
74. Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M.P.; Huang, H.; Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Republic of Korea, 4–8 June 2018; pp. 159–172.
75. Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In Proceedings of the 27th USENIX Security Symposium, Baltimore, MD, USA, 14 May 2018; pp. 1615–1631.
76. Guo, J.; Potkonjak, M. Watermarking deep neural networks for embedded systems. In Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, USA, 5–8 November 2018; pp. 1–8.
77. Le Merrer, E.; Perez, P.; Trédan, G. Adversarial frontier stitching for remote neural network watermarking. *Neural Comput. Appl.* **2020**, *32*, 9233–9244. [CrossRef]
78. Namba, R.; Sakuma, J. Robust watermarking of neural network with exponential weighting. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, Auckland, New Zealand, 7–12 July 2019; pp. 228–240.
79. Li, Z.; Hu, C.; Zhang, Y.; Guo, S. 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, San Juan, PR, USA, 9–13 December 2019; pp. 126–137.
80. Kapusta, K.; Thouvenot, V.; Bettan, O. Watermarking at the service of intellectual property rights of ML models. In Proceedings of the Actes de la conférence CAID 2020, Paris, France, 24–26 April 2020; p. 75.
81. Lounici, S.; Önen, M.; Ermis, O.; Trabelsi, S. Blindspot: Watermarking through fairness. In Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security, Chicago, IL, USA, 28–30 June 2022; pp. 39–50.
82. Kallas, K.; Furon, T. RoSe: A RObust and SEcure Black-Box DNN Watermarking. In Proceedings of the IEEE Workshop on Information Forensics and Security, Online, 12–16 December 2022.
83. Qiao, T.; Ma, Y.; Zheng, N.; Wu, H.; Chen, Y.; Xu, M.; Luo, X. A novel model watermarking for protecting generative adversarial network. *Comput. Secur.* **2023**, *127*, 103102. [CrossRef]
84. Kallas, K.; Furon, T. Mixer: DNN Watermarking using Image Mixup. In Proceedings of the ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5.
85. Hua, G.; Teoh, A.B.J.; Xiang, Y.; Jiang, H. Unambiguous and High-Fidelity Backdoor Watermarking for Deep Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–14. https://doi.org//10.1109/TNNLS.2023.3250210. [CrossRef]

86. Wang, T.; Kerschbaum, F. RIGA: Covert and Robust White-Box Watermarking of Deep Neural Networks. *arXiv* **2019**, arXiv:1910.14268.
87. Zhang, J.; Chen, D.; Liao, J.; Zhang, W.; Hua, G.; Yu, N. Passport-aware normalization for deep model protection. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22619–22628.
88. Yang, W.; Yin, Y.; Zhu, G.; Gu, H.; Fan, L.; Cao, X.; Yang, Q. FedZKP: Federated Model Ownership Verification with Zero-knowledge Proof. *arXiv* **2023**, arXiv:2305.04507.
89. Rosasco, L.; De Vito, E.; Caponnetto, A.; Piana, M.; Verri, A. Are loss functions all the same? *Neural Comput.* **2004**, *16*, 1063–1076. [CrossRef]
90. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. Technical Report. 2009. Available online: http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf (accessed on 16 August 2023).
91. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
92. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2014**, arXiv:1412.6572.
93. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 308–318.
94. Shao, S.; Yang, W.; Gu, H.; Lou, J.; Qin, Z.; Fan, L.; Yang, Q.; Ren, K. FedTracker: Furnishing Ownership Verification and Traceability for Federated Learning Model. *arXiv* **2022**, arXiv:2211.07160.
95. Liu, X.; Shao, S.; Yang, Y.; Wu, K.; Yang, W.; Fang, H. Secure federated learning model verification: A client-side backdoor triggered watermarking scheme. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, VIC, Australia, 17–20 October 2021; pp. 2414–2419.
96. Liang, J.; Wang, R. FedCIP: Federated Client Intellectual Property Protection with Traitor Tracking. *arXiv* **2023**, arXiv:2306.01356.
97. Chen, J.; Li, M.; Zheng, H. FedRight: An Effective Model Copyright Protection for Federated Learning. *arXiv* **2023**, arXiv:2303.10399.
98. Yang, W.; Shao, S.; Yang, Y.; Liu, X.; Xia, Z.; Schaefer, G.; Fang, H. Watermarking in Secure Federated Learning: A Verification Framework Based on Client-Side Backdooring. *arXiv* **2022**, arXiv:2211.07138.
99. Li, F.Q.; Wang, S.L.; Liew, A.W.C. Towards practical watermark for deep neural networks in federated learning. *arXiv* **2021**, arXiv:2105.03167.
100. Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; Zhao, B.Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 707–723.
101. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
102. Wahba, G. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. *Adv. Kernel-Methods-Support Vector Learn.* **1999**, *6*, 69–87.
103. De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3366–3385.
104. Cao, X.; Jia, J.; Gong, N.Z. 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, Virtual Event, 7–11 June 2021; pp. 14–25.
105. Becker, G. *Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis*; Techical Report; Ruhr-University Bochum: Bochum, Germany, 2008; Volume 12, p. 19.
106. Bank, D.; Koenigstein, N.; Giryes, R. Autoencoders. *arXiv* **2020**, arXiv:2003.05991.
107. Jain, A.; Krenn, S.; Pietrzak, K.; Tentes, A. Commitments and efficient zero-knowledge proofs from learning parity with noise. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, 2–6 December 2012; pp. 663–680.
108. Zheng, X.; Dong, Q.; Fu, A. WMDefense: Using Watermark to Defense Byzantine Attacks in Federated Learning. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2–5 May 2022; pp. 1–6.
109. French, R.M. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.* **1999**, *3*, 128–135. [CrossRef] [PubMed]
110. Hitaj, D.; Mancini, L.V. Have you stolen my model? evasion attacks against deep neural network watermarking techniques. *arXiv* **2018**, arXiv:1809.00615.
111. Yang, J.; Zhou, K.; Li, Y.; Liu, Z. Generalized out-of-distribution detection: A survey. *arXiv* **2021**, arXiv:2110.11334.
112. Melis, L.; Song, C.; De Cristofaro, E.; Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 13–23 May 2019; pp. 691–706.
113. Wang, B.; Chen, Y.; Jiang, H.; Zhao, Z. PPeFL: Privacy-Preserving Edge Federated Learning with Local Differential Privacy. *IEEE Internet Things J.* **2023**, *10*, 15488–15500. [CrossRef]
114. Bellafqira, R.; Coatrieux, G.; Genin, E.; Cozic, M. Secure multilayer perceptron based on homomorphic encryption. In Proceedings of the Digital Forensics and Watermarking: 17th International Workshop, IWDW 2018, Jeju Island, Republic of Korea, 22–24 October 2018; pp. 322–336.

115. Ma, J.; Naas, S.A.; Sigg, S.; Lyu, X. Privacy-preserving federated learning based on multi-key homomorphic encryption. *Int. J. Intell. Syst.* **2022**, *37*, 5880–5901. [CrossRef]
116. Jin, W.; Yao, Y.; Han, S.; Joe-Wong, C.; Ravi, S.; Avestimehr, S.; He, C. FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System. *arXiv* **2023**, arXiv:2303.10837.
117. Tolpegin, V.; Truex, S.; Gursoy, M.E.; Liu, L. Data poisoning attacks against federated learning systems. In Proceedings of the Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, 14–18 September 2020; pp. 480–501.
118. Xi, B.; Li, S.; Li, J.; Liu, H.; Liu, H.; Zhu, H. Batfl: Backdoor detection on federated learning in e-health. In Proceedings of the 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), Tokyo, Japan, 25–28 June 2021; pp. 1–10.
119. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *Proc. IEEE* **2023**, *111*, 257–276. [CrossRef]
120. Hao, S.; Zhou, Y.; Guo, Y. A brief survey on semantic segmentation with deep learning. *Neurocomputing* **2020**, *406*, 302–321. [CrossRef]
121. Fernández-Delgado, M.; Sirsat, M.S.; Cernadas, E.; Alawadi, S.; Barro, S.; Febrero-Bande, M. An extensive experimental survey of regression methods. *Neural Netw.* **2019**, *111*, 11–34. [CrossRef]
122. Fu, T.; Gao, S.; Zhao, X.; Wen, J.r.; Yan, R. Learning towards conversational AI: A survey. *AI Open* **2022**, *3*, 14–28. [CrossRef]
123. Rieke, N.; Hancox, J.; Li, W.; Milletari, F.; Roth, H.R.; Albarqouni, S.; Bakas, S.; Galtier, M.N.; Landman, B.A.; Maier-Hein, K.; et al. The future of digital health with federated learning. *NPJ Digit. Med.* **2020**, *3*, 119. [CrossRef] [PubMed]
124. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Poor, H.V. Federated learning for internet of things: A comprehensive survey. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 1622–1658. [CrossRef]