# FedTracker: Furnishing Ownership Verification and Traceability for Federated Learning Model

Shuo Shao[1*] Wenyuan Yang[2*] Hanlin Gu[3] Jian Lou[1] Zhan Qin[1†] Lixin Fan[3] Qiang Yang[4] Kui Ren[1]
[1]Zhejiang University [2]Sun Yat-sen University [3]WeBank AI Lab [4] Hong Kong University of Science and Technology

## Abstract

*Copyright protection of the Federated Learning (FL) model has become a major concern since malicious clients in FL can stealthily distribute or sell the FL model to other parties. In order to prevent such misbehavior, one must be able to catch the culprit by investigating trace evidence from the model in question. In this paper, we propose FedTracker, the first FL model protection framework that, on one hand, employs global watermarks to verify ownerships of the global model; and on the other hand, embed unique local fingerprints into respective local models to facilitate tracing the model back to the culprit. Furthermore, FedTracker introduces the intuition of Continual Learning (CL) into watermark embedding, and proposes a CL-based watermark mechanism to improve fidelity. Experimental results show that the proposed FedTracker is effective in ownership verification, traceability, fidelity, and robustness.*

## 1. Introduction

Recently, Federated Learning (FL) [25, 37] has turned into one of the most popular frameworks for distributedly training high-quality deep learning models without sharing private messages. FL is nowadays widely applied in object detection [23], medical image analysis [14, 21, 33], recommendation systems [32, 36] and so on. However, protecting the data privacy in FL makes the model publicly available to multiple participants, which might lead to the model leakage issue in FL.

Model leakage refers to that malicious clients can stealthily and personally distribute or sell the FL model, which infringes the legitimate copyright of the FL group. There are two major concerns in FL copyright protection, ownership verification and traceability. Firstly, the FL model should be considered as the intellectual property of the FL group instead of any individual. The FL group needs to prove the ownership of a suspicious model. Additionally, traceability refers to tracing the real leaker who illegally disclose the global model to third-parties outside FL community. It is a new requirement for protecting the copyright of the FL model, which to our best knowledge has not been addressed before.

Recently, DNN watermarks [1, 2, 6, 20, 26, 31] are applied to protect the copyright of the FL models. WAFFLE [30] is the first watermarking scheme designed for FL, in which the Server is responsible for embedding the *backdoor-based* watermarks into the FL model. Liu et al. [22] and Li et al. [17] propose to entitle the Clients to embed private watermarks into the global model. However, existing works only focus on ownership verification. None of them can provide traceability of the model leakers for FL models. Furthermore, the existing client-side FL watermarking scheme allowing Clients to embed watermark is tantamount to accepting all sorts of model poisoning. Thus, the Client-side watermarking scheme is not secure in practice. Existing works are not sufficient for protecting from model leakage in practical FL scenarios.

We aim to design a complete FL copyright protection framework that can furnish both ownership verification and traceability for the FL model. The challenges are three-fold. First, the server should embed watermarks to provide ownership verification. It is challenging to preserve the utility of the model in condition of a lack of natural data. Second, due to the consistency of the models received by all the clients, the models distributed to each client are the same. It is difficult to trace the leaker from the suspicious model. Last but not least, there may be conflicts between the ownership verification mechanism and the traceability mechanism. How to avoid or minimize the conflict is also challenging.

In this paper, we propose FedTracker, the first FL model protection framework furnishing both ownership verification and traceability. The illustration of the problem and our framework is shown in Figure 1. In particular, we propose a bi-level protecting mechanism to provide ownership verification and traceability respectively. On the one hand, FedTracker embeds a global watermark into the global model after aggregation in each iteration. The global watermark can verify the ownership of the model. Also, we consider

*Equal contribution.
†Corresponding author, email: qinzhan@zju.edu.cn

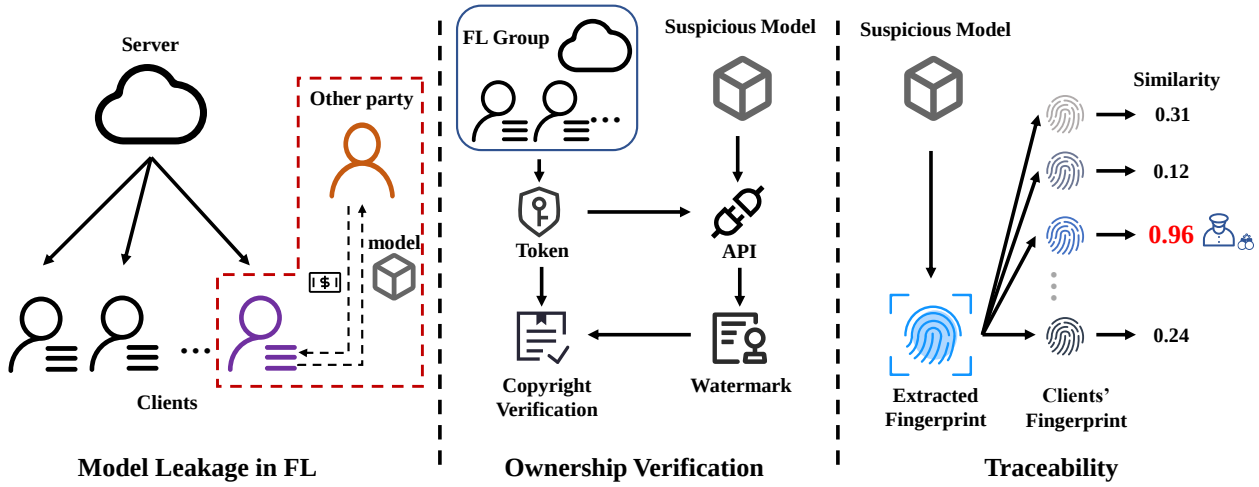


Figure 1. Illustration of model leakage in FL and the solution framework. Left: the malicious client can sell the model to other parties. Middle: FL group needs to verify the ownership through API access. Right: fingerprint is extracted from the model and compared with all the fingerprints of Clients to trace the leaker.

embedding watermark as learning data from another domain and propose a global watermark mechanism based on Continual Learning (CL). CL can help retain the utility of the model. On the other hand, before the model distribution step in each iteration, we insert a unique local fingerprint for each client to identify the clients and trace the malicious. We also propose a Fingerprint Similarity Score (FSS) to measure the similarity of the extracted fingerprint and Clients' fingerprints. Compared with tradition hamming distance, FSS can provide continuous numerical measurement and better distinguish different fingerprints.

Our main contributions can be summarized as follows:

- A novel FL model protection framework called FedTracker is proposed in this paper. The bi-level protecting mechanism in FedTracker can provide both ownership verification and traceability for FL models.
- FedTraker embeds the global watermark into the global model after aggregation to verify the ownership of the global model. Inspired by the intuition of Continual Learning, a CL-based global watermark mechanism is proposed to maintain the utility of the model on both primitive task and the watermark.
- FedTraker inserts the local fingerprint into the model of each client respectively before distribution. We also design a continuous similarity metric to compare the similarity of fingerprints and facilitate tracing the model back to the culprit.
- Empirical experiments show that FedTracker can provide effective ownership verification and traceability. FedTracker can also provide tremendous fidelity and robustness against various attacks.

## 2. Background

### 2.1. Federated Learning

In the Client-Server FL [25, 37], there are two different parties involved: Clients and Server. Clients $C = \{c_i\}_{i=1}^{K}$ are the data owners in FL where $K$ is the number of Clients. And Server is responsible for collecting the intermediate local gradients $L = \{l_i\}_{i=1}^{K}$ from Clients and aggregating them into a global model $M_g$. In each iteration $t$, Clients and Server do

1. Clients get the global model $M_g^{t-1}$ of the $t-1$ round from Server. We use the superscript $t-1$ to mark the iteration number and the subscript represents the number of clients or the global model ($g$).
2. Each Client $c_i$ copy the global model onto its local model $M_i^{t-1} \leftarrow M_g^{t-1}$ and utilizes its private dataset $D_i$ to train the local model $M_i^{t-1}$ for several epochs and calculate the local gradients $l_i^t$.
3. Clients send their local gradients to Server. Server performs the federated aggregation algorithm to aggregate the local gradients into a new global model $M_g^t$.
4. Server sends the global model $M_g^t$ to all the Clients.

### 2.2. DNN Watermarking Methods

Broadly speaking, existing DNN watermarking methods can be categorized as *parameter-based* watermarks and *backdoor-based* watermarks [28, 34].

**Parameter-based Watermarks** [4, 6, 18, 31]. *Parameter-based* watermarks decide to embed the watermarks, which can be represented as an $N$-bit string $F \in \{0,1\}^N$, directly into the parameters $W$ of the models. The embedding of the watermark can be considered a binary classification prob-

lem and the watermark can be fitted by adding a specific regularization term into the loss function.

*Parameter-based* watermarks can provide multi-bit information, but the extraction of watermarks needs white-box access to the suspicious model.

**Backdoor-based Watermarks** [1, 2, 15, 19]. *Backdoor-based* watermarks utilize backdoor attack [7, 27] to insert specific trigger set $D_T$ into the DNN model. The backdoor attack leads to misclassification when encountering samples in the trigger set. The trigger set is unique to the backdoored model, thus the owner can verify its ownership by triggering the misclassification.

*Backdoor-based* watermarks can be verified through black-box access, but *backdoor-based* watermarks are usually zero-bit. This means we can only know 'yes' or 'no' instead of a bit string representing identity.

# 3. Ownership Verification and Traceability in Federated Learning

## 3.1. Problem Formulation

Ownership verification and traceability are two major concerns in the copyright protection of the FL model. In this subsection, we give the definition of ownership verification and traceability.

**Definition 1 (Ownership verification)** *Ownership verification refers to verifying the ownership of the FL group on their model. Given a suspicious model $M$ and the token $D_T$ offered by the FL group, the ownership verification mechanism* $\texttt{Verify}(M, D_T)$ *can verify whether the model belongs to the FL group.*

**Definition 2 (Traceability)** *Traceability refers to tracing the stolen model to the malicious Client in FL. Given a suspicious model $M$ and fingerprints of clients $\{F_i\}_{i=1}^K$, traceability mechanism* $\texttt{Trace}(M, \{F_i\}_{i=1}^K)$ *can extract the fingerprint from $M$ and seek out the leaker.*

Server and Clients are the two different parties in FL. Following the prior works [3, 30], we assume that the Server is a trusted and honest party, while some Clients might be malicious. Malicious clients can copy, distribute and sell the jointly-trained model stealthily.

**Adversary's assumptions:** the adversary aims to obtain a high-performance DNN model through FL, and secretly sell the model to other parties. Moreover, the adversary can try to remove the protection mechanism in the model. We assume that the adversary has the following capabilities:

- The adversary has access to its own dataset $D_{adv}$ and the distributed model $M_{adv}^t$ in any iteration round $t$. Also, the adversary can save and copy the model $M_{adv}^t$ at any time of training.
- The adversary follows the training procedure honestly to maximize the value of the model.
- The adversary can conduct several attacks to get rid of the protection mechanism such as watermarks in the DNN model.

**Defender's assumptions:** in our scenario, the Server in FL is the defender. The defender tries to furnish ownership verification and traceability for FL models. We assume that the defender can only get the aggregated gradients in each iteration to preserve data privacy. Also, we consider the worst case that the Server has no access to any natural data [30]. The defender needs to achieve the following goals:

- **Ownership verification:** the defender should design a mechanism to successfully verify the ownership of the FL groups on FL models.
- **Traceability:** given an FL model, the defender should trace the leaker correctly.
- **Fidelity:** the protection mechanism should have negligible impact on the primitive performance of the model.
- **Robustness:** the protection mechanism should resist various attacks, such as fine-tuning, pruning and overwriting.

## 3.2. Overview of FedTracker

In this paper, we propose to conduct the protecting mechanism from Server-side due to the security issues of Client-side watermarking in Section 1. In FedTracker, a bi-level protecting mechanism is included. For the first level, a so-called global watermark is embedded into the global model so that the models in FL can contain the global watermark for ownership verification. For the second level, a unique local fingerprint is inserted into the model of each client for further traceability. In each iteration, FedTracker consists of four stages which are shown in Figure 2.

- **Stage 1.** (*Local training and aggregation*) Each Client trains their model locally. The Server collects the local gradients and aggregates them into a global model.
- **Stage 2.** (*Global watermark embedding*) The Server confirms the existence of the global watermark and embeds the global watermark into the global model. The global watermark mechanism is discussed in Section 3.3.
- **Stage 3.** (*Local fingerprints insertion*) The Server first generates $K$ copies of global model ($K$ is the number of Clients). Then, a unique local fingerprint that represents a client is inserted into each copy. The local fingerprints mechanism is discussed in Section 3.4.
- **Stage 4.** (*Model distribution*) The fingerprinted models are distributed to the Clients.

Server采用backdoor的方式嵌入watermark，并且为每个client以feature的方式嵌入fingerprint

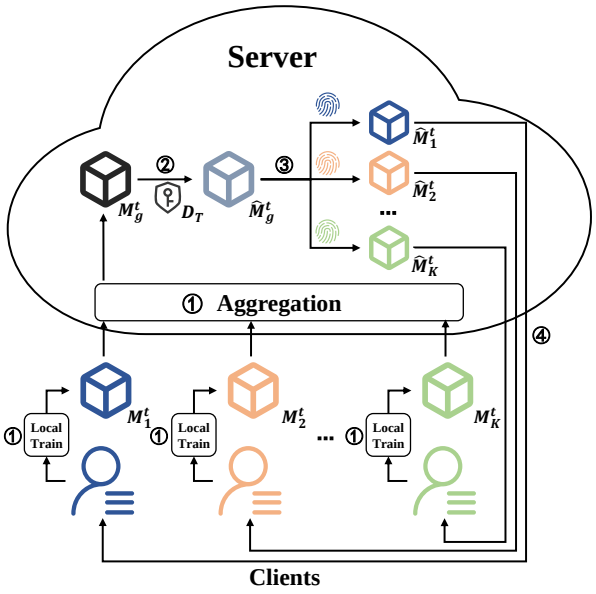


Figure 2. Workflow of FedTracker. FedTracker consists of four stages. ① Local training and aggregation. ② Global watermark embedding. ③ Local fingerprints insertion. ④ Model distribution.

### 3.3. Global Watermark Embedding

Global watermark aims to provide ownership verification for FL models. All the models with fair utility in FL should retain the global watermark. Moreover, the verification of the global watermark should be as easy as possible. In FedTracker, we take advantage of *backdoor-based* watermark and try to embed a special trigger set as the global watermark.

The global watermark mechanism can be divided into three steps.

1. (Initialization) Carefully craft a special watermark trigger set $D_T = \{(x_T, y_T)\}$ via trigger set construction methods.
2. (Trigger Set Embedding) In stage 2 discussed in Section 3.2, the Server retrains the model on the trigger set to embed the global watermark.
3. (Trigger Set Verification) When detecting a suspicious model, the FL group can take each $x_T$ in $D_T$ as input and get the output $y_T^{'}$ of the suspicious model. If most of the $y_T^{'}$ satisfy $y_T = y_T^{'}$, i.e., the accuracy of the trigger set $D_T$ is greater than a threshold $\tau$, the FL group can successfully verify their ownership on the suspicious model.

In the initialization phase, the server needs to conduct a data-free trigger set construction algorithm [30] to generate the watermark trigger set, because the server is assumed to have no access to the natural data [37]. This constraint brings a fidelity problem. The server should retrain the model on the trigger set without any natural data. It might lead to catastrophic forgetting [11] and compromise the utility of the model. It can be even worse when using models with Batch Normalization (BN) layers. To tackle this issue, FedTracker utilizes two mechanisms: (1) a CL-based global watermark mechanism. (2) Training with Frozen BN layers.

**CL-based Global Watermark Mechanism.** In FedTracker, we introduce the intuition of Continual Learning (CL) [5] to embedding watermark. Learning the trigger set can be considered as learning from another domain. Thus, CL can help to reduce the negative impact and improve the utility of the model on both two tasks.

Considering the constraints that the only messages the Server can get are the aggregated gradients, we propose a CL-based trigger set embedding algorithm which gets inspiration from Gradient Episodic Memory (GEM) [24]. Different from GEM, we directly store the gradients instead of data from prior tasks as memory. In FedTracker, we introduce *global memory*. Global memory is defined as the accumulated global gradients from the first iteration to the current iteration. Global memory $m_t$ of the $t$-th iteration can be calculated using Equation (1),

$$m_t = \sum_{j=1}^{t} g^j = \sum_{j=1}^{t} \sum_{i=1}^{K} l_i^j, \tag{1}$$

where $l_i^j$ is the local gradients of the $i$-th client in $j$-th iteration and $g^j$ is the global gradients.

The global memory $m_t$ represents the descent direction of the loss of the primitive task. While learning the trigger set task using gradient descent, we should avoid the primitive task loss increase. This can be considered as an inequality constraint measured by computing the angle between the gradients $g$ of the new task and the memory $m_t$. Mathematically, we phrase the optimization problem as:

$$\min loss(M_g^t, D^T), \quad s.t. \ \langle g, m_t \rangle \geq 0. \tag{2}$$

If the constraint is satisfied, we can directly apply the gradients $g$ to optimize the loss of watermark without increasing the loss of the primitive task. However, if the violation occurs, we should project the gradients $g$ to the closest gradient $\tilde{g}$ satisfying the constraint. The projection problem can be phased as Equation (3).

$$\arg\min_{\tilde{g}} \|g - \tilde{g}\|_2^2, \quad s.t. \ \langle \tilde{g}, m_t \rangle \geq 0. \tag{3}$$

According to GEM [24], the problem can be solved via Quadratic Programming. The pseudocode is shown in Algorithm 1 line 11-19.

**Training with Frozen BN Layers.** As Algorithm 1 demonstrates, we freeze the Batch Normalization (BN) layers in the model during learning the trigger set task on account of the following two reasons.



用continual leanring中projection的思路防止嵌入watermark后对原任务造成遗忘

1. Since the Server has no access to the training data, the trigger set generated by the data-free trigger set construction method [1, 30] is out of the distribution of natural data. Directly learning the trigger set can significantly affect the stability of BN layers and decrease the utility. Thus freezing the BN layers can help retain the utility of the model.
2. Freezing the BN layers can also avoid negative influence on local fingerprints which will be discussed in Section 3.4.

### 3.4. Local Fingerprints Insertion

Local Fingerprints aim to identify the models of clients and trace the model leaker. In FedTracker, we utilize multi-bit *parameter-based* method as local fingerprints. A unique local fingerprint is inserted into the model of each client.

**Fingerprints Inserting.** In FedTracker, given a weight matrix $W \in \mathbb{R}^M$ of the model, we use the following equation to represent the local fingerprint $F \in \{-1, 1\}^N$

$$sgn(AW) = F, \tag{4}$$

where $A$ is the secret key matrix that is randomly generated by the Server. In FedTracker, one secret key matrix $A_i \in \mathbb{R}^{M \times N}$ is generated for each fingerprint $F_i \in \{-1, 1\}^N$. $sgn(\cdot)$ originates from the sign function.

$$sgn(x) = \begin{cases} -1, x < 0 \\ 1, x \geq 0 \end{cases}. \tag{5}$$

In FedTracker, we choose the weight of the BN layers to insert local fingerprints, and it has been reported that the fingerprints in BN layers can achieve great robustness [6]. Recall that in BN layers, we calculate the following equation [10]

$$o = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{6}$$

where $\mu$ and $\sigma$ are the mean and variance of input batch $x$, $\epsilon$ is a small constant. $\gamma$ and $\beta$ are learnable parameters. We choose $\gamma$ to embed the local fingerprint. $\gamma$ from different BN layers are concatenated into one vector $W^\gamma$. We define the loss function $L_f(W_i^\gamma)$ of fitting the fingerprint for the $i$-th client as Equation (7).

用BN层的系数插入fingerprint

$$L_f(W_i^\gamma) = \mathrm{HL}(W_i^\gamma, A_i, F_i) = \sum_{j=1}^{|F_i|} \max(\delta - b_{ij} f_{ij}, 0). \tag{7}$$

In Equation (7), $b_{ij}$ is the $j$-th bit in $B_i = A_i W_i^\gamma$, while $f_{ij}$ is the $j$-th bit in $F_i$. $\delta$ is the hyperparameter controlling the robustness of the fingerprint. We optimize $L_f$ using gradient descent with a small learning rate to reduce the impact on other tasks. The pseudocode of inserting fingerprints is demonstrated in Algorithm 1, line 20-23.

**Algorithm 1** FedTracker Procedure

**Input:** Clients datasets $\{D_i\}_{i=1}^K$, watermark trigger set $D_T$, local fingerprints $\{F_i\}_{i=1}^K$, secret key matrices $\{A_i\}_{i=1}^K$.
**Output:** clients' models $\{M_i^T\}_{i=1}^K$, the global model $M_g^T$.

1: $M_g^0 \leftarrow Initialize()$
2: **for** $i = 1$ to $K$ **do**
3: $\quad M_i^0 \leftarrow M_g^0$
4: $m \leftarrow 0$
5: **for** $t = 1$ to $T$ **do**
6: $\quad$ // ① Local training and aggregation
7: $\quad$ **for** $i = 1$ to $K$ **do**
8: $\quad\quad M_i^t \leftarrow LocalTrain(M_i^{t-1}, D_i)$
9: $\quad\quad l_i^t \leftarrow M_i^{t-1} - M_i^t$
10: $\quad M_g^t \leftarrow \frac{1}{K} \sum_{i=1}^K M_i^t$
11: $\quad$ // ② Global watermark embedding
12: $\quad m \leftarrow m + \frac{1}{K} \sum_{i=1}^K l_i^t$ // Update memory
13: $\quad FreezeBN(M_g^t)$
14: $\quad g \leftarrow CalGrad(M_g^t, D_T)$
15: $\quad$ **if** $\langle g, m \rangle < 0$ **then**
16: $\quad\quad \tilde{g} \leftarrow Project(g, m)$ 将梯度投影防止遗忘
17: $\quad$ **else**
18: $\quad\quad \tilde{g} \leftarrow g$
19: $\quad \hat{M}_g^t \leftarrow M_g^t - \lambda \tilde{g}$
20: $\quad$ // ③ Local fingerprints insertion
21: $\quad$ **for** $i = 1$ to $K$ **do**
22: $\quad\quad M_i^t \leftarrow Copy(\hat{M}_g^t)$
23: $\quad\quad M_i^t \leftarrow InsertFingerprint(M_i^t, A_i, F_i)$
24: **return** $\{M_i^T\}_{i=1}^K, M_g^T$ // ④ Model distribution

**Fingerprints Similarity Score.** After inserting the local fingerprints, it is necessary to define a metric to measure the similarity between the extracted fingerprint and the Clients' fingerprints. In FedTracker, we propose Fingerprint Similarity Score:

**Definition 3 (Fingerprint Similarity Score)** *Given the secret key matrix $A_i$, the weight matrix $W$, the fingerprint $F_i$ and $B_i = A_i W$, the Fingerprint Similarity Score (FSS) for the $i$-th client can be calculated with Equation (8).*

$$\mathrm{FSS}(A_i, F_i, W) = \mathrm{FSS}(B_i, F_i) = \sum_{j=1}^{|F_i|} \min(\delta, b_{ij} f_{ij}). \tag{8}$$

Compared with hamming distance, FSS can provide continuous metric and better robustness. While tracing the leaker, we calculate the FSS for every $A_i$ and $F_i$, and get a similarity vector. The Client with the maximal FSS is suspected of being the leaker.

**Avoid Overfitting on Fingerprints.** In FedTracker, we should avoid the situation that the model simultaneously

| Model | Number of Clients | | | |
|---|---|---|---|---|
| | 10 | 20 | 30 | 50 |
| CNN-4 | 84.70 | 87.80 | 80.70 | 86.90 |
| AlexNet | 98.90 | 96.45 | 86.83 | 82.68 |
| VGG-16 | 99.30 | 99.90 | 99.87 | 99.12 |
| ResNet-18 | 97.95 | 95.86 | 79.02 | 78.82 |

Table 1. Watermark accuracy (%) with different numbers of clients.

overfits on two fingerprints, i.e., for one $W$, there are two fingerprints $F_\mu, F_\nu$ having the same maximal FSS. This can lead to ambiguity. Li et al. [17] has proven that there exists a $W$ containing all the fingerprints if $KN \leq M$. $N$ and $M$ are the dimensions of $F$ and $W$. Therefore, while inserting the local fingerprints, we should follow $KN > M$.

FedTracker also utilizes early-stopping technique to prevent the situation. When the FSS is higher than a threshold, FedTraker will stop inserting the fingerprint which can reduce the influence of inserting fingerprint on the utility of model.

# 4. Experiments

## 4.1. Experiment Settings

**Datasets and Models.** In our experiments, we use four different Convolution Neural Networks (CNN) to learn three different datasets including MNIST [35], CIFAR10 [12] and TinyImagenet [16]. For MNIST, we utilize a four-layer CNN (denoted as CNN-4) which is also used in [25]. For CIFAR10, two popular models, AlexNet [13] and VGG-16 [29], are chosen. ResNet-18 [9] is trained on TinyImagenet, a larger dataset with 200 classes. The first three models are slightly modified by adding a BN layer after each convolution layer.

**FL Settings.** To simulate the FL scenario, we set the default number of clients to 50. In each iteration, we choose 40% clients to train their local models for 2 local epochs. The learning rate of clients is 0.01. For the default setting, we assume that the distribution of the clients' data follows independent identical distribution (i.i.d), which means the training dataset is uniformly allocated to each client.

**Watermark and Fingerprint.** For global watermark, we implement WafflePattern [30] to generate the trigger set. Each class has 10 samples in the trigger set. The learning rate is set to 0.005 while embedding global watermarks. For local fingerprints, we set the default length of fingerprints to 128, which is sufficient for tracing 50 clients. The learning rate is 0.05.

**Metrics.** For ownership verification, we use the accuracy of the trigger set, named watermark accuracy, as the metric. For traceability, we define another metric named *Traceability Rate*. Traceability Rate (TR) is defined as the rate of that the client with the largest FSS happens to be the holder of the model.

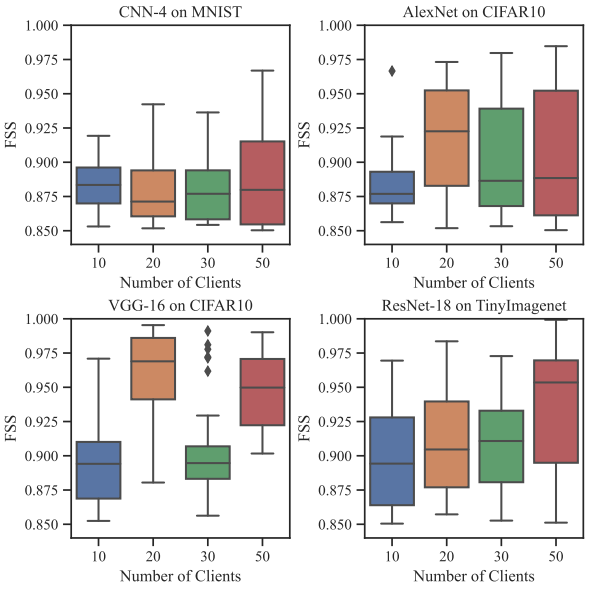


Figure 3. FSS evaluation with different numbers of clients.

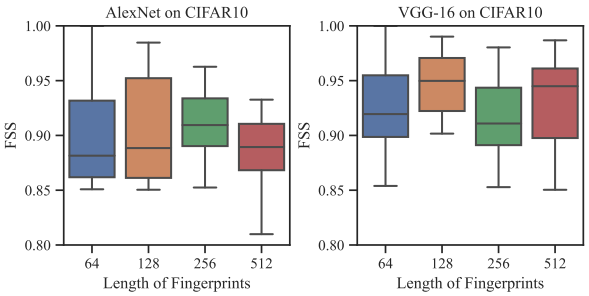


Figure 4. FSS evaluation with different lengths of local fingerprints.

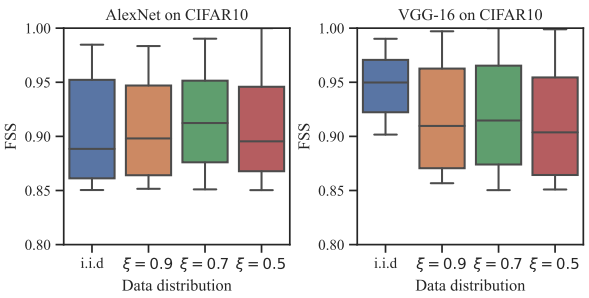


Figure 5. FSS evaluation under non-i.i.d settings.

## 4.2. Effectiveness

**Effectiveness with Different Numbers of Clients.** We implement FedTracker with four different numbers of clients in FL, 10, 20, 30, 50. For traceability, we achieve 100% TR in all experiments. The FSS results are shown in Figure 3. In all experiments, the minimal FSS is higher than 0.85,

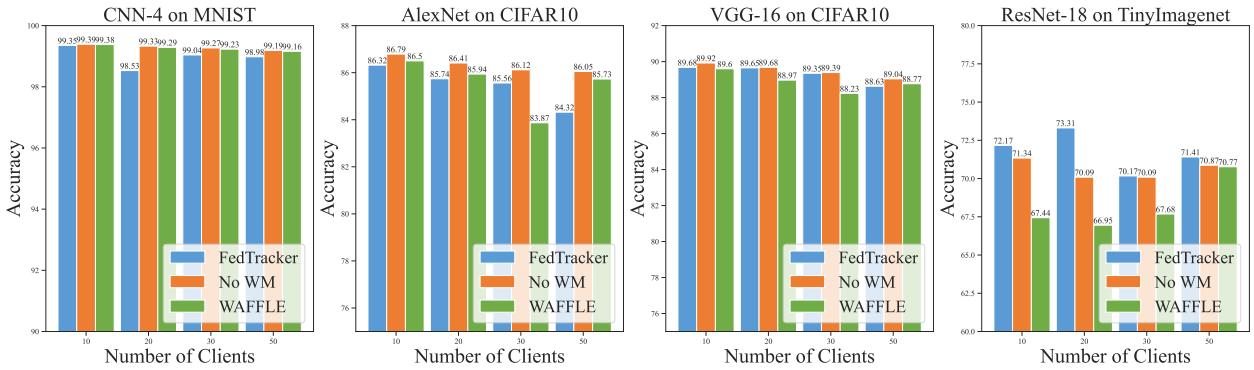


Figure 6. Accuracy on main task of FedTracker comparing with model without watermark (No WM) and WAFFLE.

| Model | Size of watermark trigger set | | | |
|---|---|---|---|---|
| | 100 | 200 | 500 | 1000 |
| AlexNet | 82.68 | 77.26 | 81.86 | 79.89 |
| VGG-16 | 99.12 | 99.23 | 97.52 | 88.50 |

Table 2. Watermark accuracy (%) with different sizes of watermark trigger set.

| Model | Data distribution | | | |
|---|---|---|---|---|
| | i.i.d | $\xi = 0.9$ | $\xi = 0.7$ | $\xi = 0.5$ |
| AlexNet | 82.68 | 81.90 | 80.24 | 79.50 |
| VGG-16 | 99.12 | 99.14 | 98.40 | 98.88 |

Table 3. Watermark accuracy (%) under non-i.i.d settings.

which indicates a successful insertion of local fingerprints.

For ownership verification, the results are shown in Table 1. As the number of clients grows, the watermark accuracies decrease but are still over $82\%$ for the first three models. For ResNet-18 on the TinyImagenet task, considering the number of classes is 200, the least $78.82\%$ accuracy is still high in this task.

**Effectiveness with Different Lengths of Local Fingerprints.** In this experiment, we test FedTracker with the fingerprint length of $64, 128, 256, 512$. The number of Clients is fixed at 50, which is the most complicated case in our experiments. We still achieve $100\%$ TR, and the FSS evaluations are shown in Figure 4. Experimental results demonstrate that the length of the local fingerprints does not significantly affect the effectiveness of FedTracker, since the minimal FSS is greater than $0.80$.

**Effectiveness with Different Sizes of Watermark Trigger Set.** In this experiment, we choose different sizes of the watermark trigger set to examine the effectiveness. The results in Table 2 demonstrate that although the watermark accuracy drops slightly with the increase of sizes, the watermark still retains in the FL models. It proves that FedTracker is effective despite the sizes of the trigger set.

**Effectiveness under Non-i.i.d Settings.** We also measure the effectiveness of FedTracker under non-i.i.d settings, which is a common problem in FL. Following prior works [38], we utilize Dirichlet distribution to generate non-i.i.d data for clients. The concentration parameter $\xi \in [0, 1]$ of Dirichlet distribution controls the degree of non-i.i.d. We conduct the experiments with $\xi = 0.5, 0.7, 0.9$. The results are demonstrated in Figure 5 and Table 3. We can see that the non-i.i.d situation has a slight impact on FedTracker since the FSS and watermark accuracies drop. But it is not sufficient to make FedTracker ineffective.

| Model | Method | Number of Clients | | | |
|---|---|---|---|---|---|
| | | 10 | 20 | 30 | 50 |
| AlexNet | WAFFLE | 86.50 | 85.94 | 83.87 | 85.73 |
| | FedTracker | **86.99** | **86.56** | **85.82** | **85.98** |
| VGG-16 | WAFFLE | 89.60 | 88.97 | 88.23 | **88.77** |
| | FedTracker | **89.99** | **89.41** | **89.31** | 88.68 |

Table 4. Accuracy (%) on main task using WAFFLE or FedTracker to embed watermark.

### 4.3. Fidelity

In the fidelity experiments, we compare the model's utility on the primitive task in FedTracker and the other two cases. We first train the model without any watermark and fingerprints. We also implement WAFFLE [30] as the second baseline method. The accuracies on the classification tasks are shown in Figure 6. From Figure 6, FedTracker does not have a significant drop in classification accuracies compared with the model without any watermark and fingerprint. In most cases, the utility difference between the FedTracker model and the model without watermarks is below $0.5\%$. Moreover, compared with WAFFLE, FedTracker achieves nearly the same utility. However, FedTracker can provide traceability which is not a concern in WAFFLE.

Furthermore, we test the fidelity performance with or without our proposed CL-based watermark embedding algorithm. In this experiment, we do not insert local fingerprints to test our proposed CL-based algorithm. The results are in Table 4. In 7 of 8 cases, our proposed CL-based algorithm achieves better utility than WAFFLE, which demonstrates that FedTracker can improve fidelity.

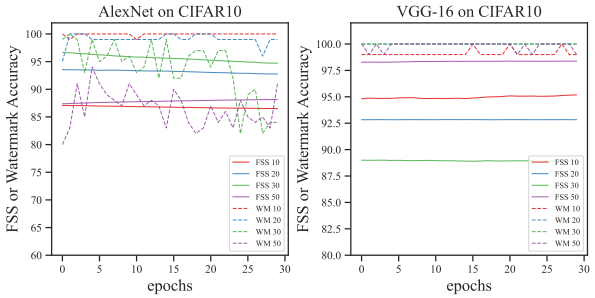


Figure 7. FSS and watermarking accuracy (WM) against fine-tuning attack. The number behind refers to the number of Clients in FL.

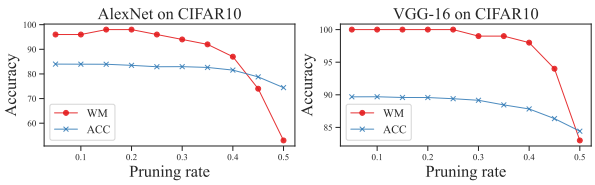


Figure 8. Watermark accuracy (WM) and classification accuracy (ACC) against pruning attack which does not prune the BN layers.

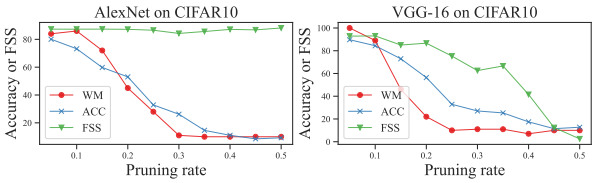


Figure 9. FSS, watermark accuracy (WM) and classification accuracy (ACC) against pruning attack which prunes the BN layers.

### 4.4. Robustness

**Robustness against Fine-tuning Attack.** Fine-tuning attack refers to training the model with the private dataset of the malicious client for several epochs. We randomly choose a client to conduct a 30-epoch fine-tuning. In Figure 7, the FSS metric rarely changes in the 30 epochs. While the watermark accuracy fluctuates but remains a high value in general, which indicates the robustness of FedTracker against fine-tuning.

**Robustness against Pruning Attack.** We evaluate the robustness of our watermark and fingerprints against pruning attack, especially parameter pruning [8]. In parameter pruning, which is also conducted in [31], we prune units in the model by zeroing out the ones with the lowest $l_1$-norm. We divide the pruning attack into two cases: (1) pruning all the layers of the model except the BN layers; (2) pruning the BN layers.

In the first case, pruning other layers do not affect the fingerprints in BN layers. Thus, the watermark is the only one that might be influenced. The watermark accuracies and classification accuracies after pruning are shown in Figure 8. As the pruning rate increases, the watermark accuracy and classification accuracy decrease simultaneously. In most cases, the watermark accuracy is above $80\%$. The only exception is pruning $50\%$ neurons of AlexNet. However, the classification accuracy of the pruned model has nearly $10\%$ decrease, which means the model has lost its value. Thus, we do not think it is a successful attack.

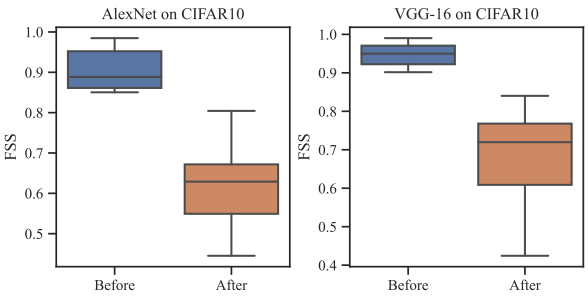


Figure 10. FSS before and after overwriting attack.

In the second case, pruning BN layers can affect both watermarks and fingerprints. From Figure 9, it can be observed that the classification accuracy drops significantly when pruning the BN layers. When the model still retains its utility, the watermark accuracy and FSS are still high enough to provide ownership verification and traceability. In Conclusion, in the two cases, the pruning attack cannot successfully remove the watermark and fingerprints in the FL model, which demonstrates the robustness of FedTracker against the pruning attack.

**Robustness against Overwriting Attack.** The malicious client might randomly generate a fingerprint $F_m$ and the secret key matrix $A_m$, and try to overwrite the fingerprint in the model. From Figure 10, the FSS metrics drop after the overwriting attack. However, the TR of the $50$ models is still $100\%$, which means FedTracker can still trace the leaker successfully. The result demonstrates that although the overwriting attack has an impact on the fingerprints, it has little effect on the relative magnitudes of FSS. FedTracker remains effective against the overwriting attack.

## 5. Conclusion

In this paper, we propose FedTracker, the first FL model protection framework to simultaneously furnish ownership verification and traceability. Global watermark and local fingerprints mechanisms are utilized to verifying the ownership of the FL model and trace the model back to the leaker respectively. We accordingly design a CL-based watermark mechanism to improve fidelity and a novel metric named Fingerprint Similarity Score to measure the similarity of fingerprints. Evaluations of various tasks and settings show that FedTraker is effective in ownership verification, traceability, fidelity, and robustness.

## References


[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of 2018 USENIX Security Symposium*, pages 1615–1631, 2018. 1, 3, 5

[2] Arpit Bansal, Ping-yeh Chiang, Michael Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified Neural Network Watermarks with Randomized Smoothing. In *Proceedings of the 2022 International Conference on Machine Learning*, pages 1450–1465. PMLR, July 2022. 1, 3

[3] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017. 3

[4] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of 2019 International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 485–497, 2019. 2

[5] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 4

[6] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: embedding passports to defeat ambiguity attacks. In *Proceedings of 2019 International Conference on Neural Information Processing Systems*, pages 4714–4723, 2019. 1, 2, 5

[7] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 3

[8] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28, 2015. 8

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 5

[11] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 4

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech. Rep.*, 2009. 6

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 6

[14] Rajesh Kumar, Abdullah Aman Khan, Jay Kumar, A Zakria, Noorbakhsh Amiri Golilarz, Simin Zhang, Yang Ting, Chengyu Zheng, and WenYong Wang. Blockchain-federated-learning and deep learning models for covid-19 detection using CT imaging. *IEEE Sensors Journal*, 2021. 1

[15] Yingjie Lao, Peng Yang, Weijie Zhao, and Ping Li. Identification for Deep Neural Network: Simply Adjusting Few Weights! In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1328–1341, Kuala Lumpur, Malaysia, May 2022. IEEE. 3

[16] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 6

[17] Bowen Li, Lixin Fan, Hanlin Gu, Jie Li, and Qiang Yang. FedIPR: Ownership Verification for Federated Deep Neural Network Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, Mar. 2022. 1, 6

[18] Guobiao Li, Sheng Li, Zhenxing Qian, and Xinpeng Zhang. Encryption resistant deep neural network watermarking. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3064–3068. IEEE, 2022. 2

[19] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In *Proceedings of 2019 Annual Computer Security Applications Conference*, pages 126–137, 2019. 3

[20] Jian Han Lim, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protect, show, attend and tell: Empowering image captioning models with ownership protection. *Pattern Recognition*, 122:108285, Feb. 2022. 1

[21] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1013–1023, 2021. 1

[22] Xiyao Liu, Shuo Shao, Yue Yang, Kangming Wu, Wenyuan Yang, and Hui Fang. Secure Federated Learning Model Verification: A Client-side Backdoor Triggered Watermarking Scheme. In *Proceedings of 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2414–2419, Melbourne, Australia, Oct. 2021. IEEE. 1

[23] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13172–13179, 2020. 1

[24] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017. 4

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of 2017 International Conference Artificial Intelligence and Statistics*, pages 1273–1282, 2017. 1, 2, 6

[26] Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protecting Intellectual Property of Generative Adversarial Networks from Ambiguity Attacks. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3629–3638, Nashville, TN, USA, June 2021. IEEE. 1

[27] Xiangyu Qi, Tinghao Xie, Ruizhe Pan, Jifeng Zhu, Yong Yang, and Kai Bu. Towards practical deployment-stage backdoor attack on deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13347–13357, 2022. 3

[28] Francesco Regazzoni, Paolo Palmieri, Fethulah Smailbegovic, Rosario Cammarota, and Ilia Polian. Protecting artificial intelligence IPs: A survey of watermarking and fingerprinting for machine learning. *CAAI Transactions on Intelligence Technology*, 6(2):180–191, 2021. 2

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[30] Buse GA Tekgul, Yuxi Xia, Samuel Marchal, and N Asokan. WAFFLE: Watermarking in federated learning. In *Proceedings of 2021 International Symposium on Reliable Distributed Systems*, pages 310–320, 2021. 1, 3, 4, 5, 6, 7

[31] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of 2017 ACM International Conference on Multimedia Retrieval*, pages 269–277, 2017. 1, 2, 8

[32] Omar Abdel Wahab, Gaith Rjoub, Jamal Bentahar, and Robin Cohen. Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems. *Information Sciences*, 601:189–206, 2022. 1

[33] An Xu, Wenqi Li, Pengfei Guo, Dong Yang, Holger R. Roth, Ali Hatamizadeh, Can Zhao, Daguang Xu, Heng Huang, and Ziyue Xu. Closing the generalization gap of cross-silo federated medical image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20866–20875, June 2022. 1

[34] Mingfu Xue, Yushu Zhang, Jian Wang, and Weiqiang Liu. Intellectual Property Protection for Deep Learning Models: Taxonomy, Methods, Attacks, and Evaluations. *IEEE Transactions on Artificial Intelligence*, pages 1–1, 2021. 2

[35] LeCun Y, Cortes C, and Burges C. MNIST handwritten digit database, 2010. 6

[36] Liu Yang, Ben Tan, Vincent W Zheng, Kai Chen, and Qiang Yang. Federated recommendation systems. In *Federated Learning*, pages 225–239. Springer, 2020. 1

[37] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019. 1, 2, 4

[38] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *Proceedings of 2019 International Conference on Machine Learning*, pages 7252–7261, 2019. 7