Hindawi
Complexity
Volume 2021, Article ID 8261663, 20 pages
https://doi.org/10.1155/2021/8261663

Research Article

Federated Learning: A Distributed Shared Machine Learning Method

Kai Hu,[1,2] **Yaogen Li**,[1] **Min Xia**,[1,2] **Jiasheng Wu**,[1] **Meixia Lu**,[1] **Shuai Zhang**,[1] **and Liguo Weng**[1,2]

[1]*Nanjing University of Information Science & Technology, Nanjing 210044, China*
[2]*Jiangsu Provincial Collaborative Innovation Center for Atmospheric Environment and Equipment Technology, Nanjing University of Information Science & Technology, Jiangsu, Nanjing 210044, China*

Correspondence should be addressed to Min Xia; xiamin@nuist.edu.cn

Received 16 June 2021; Revised 13 August 2021; Accepted 16 August 2021; Published 30 August 2021

Academic Editor: Siew Ann Cheong

Federated learning (FL) is a distributed machine learning (ML) framework. In FL, multiple clients collaborate to solve traditional distributed ML problems under the coordination of the central server without sharing their local private data with others. This paper mainly sorts out FLs based on machine learning and deep learning. First of all, this paper introduces the development process, definition, architecture, and classification of FL and explains the concept of FL by comparing it with traditional distributed learning. Then, it describes typical problems of FL that need to be solved. On the basis of classical FL algorithms, several federated machine learning algorithms are briefly introduced, with emphasis on deep learning and classification and comparisons of those algorithms are carried out. Finally, this paper discusses possible future developments of FL based on deep learning.

1. Introduction

In the era of big data, people pay more and more attention to data security and user's privacy; protection of data has become the focus of enterprises and individuals. In addition, data leakage has attracted the attention of governments and public media in recent years. The world major powers and major unions have enforced the supervision of citizens' data security and privacy in law; the General Data Protection Regulations (GDPR) issued by the European Union [1] have come into effect on May 25, 2018. China's Cyber Security Law, promulgated in 2017, requires Internet companies not to disclose or tamper with the personal information they collect from users, and when conducting data transactions with third parties, they need to ensure that both Internet company and third party comply with user data protection obligations [2–4]. The protection of data privacy in various countries becomes stricter, which makes large-scale user private data transfers between different companies in the future no longer allowed. The promulgation of these laws and regulations, on the one hand, protects the privacy of users and, on the other hand, prohibits big data from being excavated arbitrarily, which restricts the development of artificial intelligence. Big data is the basis of large-scale distributed ML. Under the restrictions of the above-mentioned laws and regulations, data often exist in the form of isolated islands among different enterprises; even among different subsidiaries of one group.

The term "federated learning" was put forward by McMahan et al. [5] in 2016: "We call our approach FL because learning tasks are solved through a loose federated of participating devices (what we call clients) coordinated by a central server." FL was originally defined as a distributed ML method that uses multiple user data to train a central model [6]. The purpose of FL is to carry out efficient distributed ML between multiparticipants or multicomputing nodes on the premise of ensuring the information security of big data exchange, protecting mobile data and personal privacy, and ensuring legal compliance. FL uses the framework of classical distributed ML and adopts distributed ML technology, but the control of the central server is different from that of distributed ML. Researchers can mine and utilize data without violating laws and regulations. In a broad sense, FL refers to a method where the data owner can realize the training of models without uploading local data [2]. The modeling of FL is based on the local model uploaded by each participant, and then the joint training model is returned to each participant to get similar results to traditional ML without violating laws; this makes FL have the advantage of confidentiality.

However, the classical algorithm of FL has some shortcomings in dealing with nonindependent and identically distributed data, communication transmission, and model establishment, and their resulting solution is too numerous to enumerate. Therefore, after consulting the relevant literature, this paper introduces classical FL and FL algorithms that have been promoted in some aspects. Moreover, in the era of big data, the effect of FL based on deep learning is more effective. This paper focuses on recent developments of federated deep learning algorithms; they are sorted out and summarized. We hope this article can make it easier for readers to quickly review the whole FL field, especially federated deep learning subfield.

The content of this paper includes the following: Section 2 introduces the basic knowledge of FL; Section 3 introduces some unsolved problems of FL; Section 4 introduces the FL algorithm based on ML; Section 5 introduces the FL algorithm based on pan-deep learning; Section 6 introduces the attack of FL; Section 7 describes future challenges of FL; and finally, the 8th summarizes this paper.

2. Basic Knowledge of Federated Learning

2.1. Machine Learning. With the rapid development of ML, its models are becoming more and more complex and effective [7, 8]. The core idea of ML is that the computer learns the mapping between input and output according to existing data samples: f_w: $x \longrightarrow y$, where x is the input, y is the output, f is the corresponding rule, and$w$ is the parameter to be learned. According to the corresponding relationship, the model predicts the output value of the next input. The purpose of ML is to make the gap between the predicted value and the real value as small as possible. The mathematics is expressed as

$$\arg\min_{w} L(x, y, w) = \left\| f_w(x) - y \right\|. \quad (1)$$

In traditional ML, such as backpropagation neural network (BPNN) and convolutional neural networks (CNNs), the learning process of this parameter is all concentrated on one computer, and the commonly used methods are gradient descent and a series of improved algorithms. The core algorithm of FL is very similar to the Stochastic Gradient Descent (SGD) method [7]. In SGD, a sample is randomly selected from all samples to participate in the operation at each iteration.

2.2. Distributed Machine Learning. Distributed machine learning combines multiple computers for computing. Its core goal is to disassemble computing tasks into multiple small tasks and perform computing on multiple local processors. Its final training requires a central server to deal with the data uploaded by local clients; as a result, communication and privacy security is difficult to be guaranteed. Algorithm 1 shows the distributed machine learning algorithm.

2.3. Federated Learning. FL is different from distributed ML; in FL, the information uploaded by each participant to the server is no longer the original data, but a trained submodel. At the same time, the FL also allows asynchronous transmission [9], and the communication requirements can be appropriately reduced. On this basis, the formula of federated machine learning can be updated as follows:

$$\arg\min_{w} L(x, y, w) = \sum_{k} p_k L_k(x, y, w), \quad (2)$$

where k is the number of clients, p_k is the weight value of the kth client, and the scenario for FL is the decentralized multiuser $\{F_1, F_2, \ldots, F_k\}$. Each client user has the current user's data set $\{D_1, D_2, \ldots, D_k\}$. In deep learning, these data are sorted out into a data set $D = U_1 \cup U_2 \cup \cdots \cup U_k$. The practice of FL is no longer to simply aggregate them to form a new data set to complete the next stage of training tasks. Suppose the global model after the completion of a federal modeling task is M_{FED} and the corresponding training model after aggregation is M_{SUM}. Generally speaking, the global model M_{FED} is functioning due to the operation of parameter exchange and aggregation. There will be a loss of accuracy during the entire training process; that is, the performance of the global model M_{FED} is not as good as the performance of the aggregate model M_{SUM}. To quantify this difference, we define the performance of the global model M_{FED} on the test set as V_{FED}, and the performance of the aggregate model M_{SUM} on the test set as V_{SUM}. At this time, the δ-loss accuracy [10] of the model is defined as

$$\left| V_{\text{FED}} - V_{\text{SUM}} \right| < \delta, \quad (3)$$

where δ is a nonnegative number. However, in actual situations, the aggregation model M_{SUM} cannot be obtained in the end, because the basic requirement of FL is privacy protection. According to Professor Yang's book "Federated Learning" [10], the federated average algorithm of FL can be expressed in Algorithm 2.

2.4. Federated Learning Classification. In FL, data are distributed among the participants in the form of isolated islands, and each participant can use a matrix to represent its own data. At present, according to the distribution of data feature space and sample ID space, researchers divide FL into three categories: horizontal federated learning, vertical federated learning, and federated transfer learning [11, 12].

2.4.1. Horizontal Federated Learning. Horizontal federated learning is similar to traditional distributed ML. It aims at the overlapping data characteristics of each client in FL. That

```
Distributed Machine Learning
Server side
(1) Input: data sample $X$, Y, initial model parameters $\mathbf{w}_0$, iterative step $\mu$;
(2) Divide $X$ Perry Y into collections in units of records $\{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^\mathbf{m}\}, \{\mathbf{Y}^1, \mathbf{Y}^1, \ldots, \mathbf{Y}^\mathbf{m}\}$. $m$ indicates the number of clients;
(3) Send $\mathbf{X}^\mathbf{k}, \mathbf{Y}^\mathbf{k}$ to the client $\mathbf{k}$;
(4) Execute $t$ times ($\mathbf{t} \geq 1$) for each iteration: send $\mathbf{w}_{\mathbf{t}-1}$ to the client;
(5) Receive gradient update $\mathbf{g}_\mathbf{t}^\mathbf{k}$ from client $\mathbf{k}$; execute $\mathbf{w}_\mathbf{t} \leftarrow \mathbf{w}_{\mathbf{t}-1} - \mu \sum_{\mathbf{i}=1}^{\mathbf{m}} \mathbf{g}_\mathbf{t}^\mathbf{k}$;
(6) To determine whether the termination condition is met: if so, it will be terminated; otherwise, it will be executed $\mathbf{t} \leftarrow \mathbf{t} + 1$;
Client side k
(1) Input: $\mathbf{X}^\mathbf{k}, \mathbf{Y}^\mathbf{k}, \mathbf{w}_{\mathbf{t}-1}$;
(2) Batches are randomly selected from $\mathbf{X}^\mathbf{k}, \mathbf{Y}^\mathbf{k}$ records as training data $\mathbf{X}_\mathbf{b}^\mathbf{k}, \mathbf{Y}_\mathbf{b}^\mathbf{k}$;
(3) Calculated gradient $\mathbf{g}_\mathbf{t}^\mathbf{k} \leftarrow \nabla \mathbf{L}(\mathbf{X}_\mathbf{b}^\mathbf{k}, \mathbf{Y}_\mathbf{b}^\mathbf{k}, \mathbf{w}_{\mathbf{t}-1})$;
(4) Send $\mathbf{g}_\mathbf{t}^\mathbf{k}$ to server side.
```

Algorithm 1: Distributed machine learning algorithm.

```
     Federated average algorithm.
 (1) Execute in the coordinator:
 (2) Initialize $w_0$ and broadcast the original model parameter $w_0$ to all participants;
 (3) For each global model update round $t = 1, 2, ..., $ do;
 (4) The coordinator determines $C_t$, that is, determines the set of $\max(k_p, 1)$ randomly selected participants;
 (5)   For each participant $k \in C_t$ do in parallel;
 (6) Update the model parameters locally: $w_{t+1}^{(k)} \leftarrow$ participants update $(k, \overline{w_t})$ (see line 13);
 (7) Send the updated model parameter $w_{t+1}^{(k)}$ to the coordinator;
 (8)   end for
 (9) The coordinator aggregates the received model parameters, that is, using a weighted average for the received model parameters:
     $\overline{w_{t+1}} \leftarrow \sum_{k=1}^{k} (n_k/n) w_{t+1}^{(k)}$
(10) The coordinator checks whether the model parameters have converged. If it converges, the coordinator sends a signal to all
     participants to suspend model training;
(11) The coordinator broadcasts the aggregated model parameter $\overline{w_{t+1}}$ parameter-to all participants;
(12) end for
(13) Update in the participant $(\mathbf{k}, \overline{\mathbf{w_t}})$(participants $\mathbf{k}$, $\forall \mathbf{k} = 1, 2, \ldots, \mathbf{K}$ are executed in parallel)
(14) Get the latest model parameters from the server, that is, set $w_{1,1}^{(k)} = \overline{w_t}$;
(15) For each local iteration from 1 to the number of iterations S $i$ do;
(16) Batches $\leftarrow$ randomly divide the data set $D_k$ into the size of the batch $M$;
(17) Obtain the local model parameters from the previous iteration, set $w_{1,i}^{(k)} = w_{B,i-1}^{(k)}$;
(18)   For batch number $b$ do from 1 to batch quantity $B = n_k/M$;
(19) Calculate batch gradient $g_k^{(b)}$;
(20) Update model parameters locally: $w_{b+1,i}^{(k)} \leftarrow w_{b,i}^{(k)} - \eta g_k^{(b)}$;
(21)   end for
(22) end for
(23) Get the local model parameter update $w_{t+1}^{(k)} = w_{B,S}^{(k)}$, and send it to the coordinator (for participants of $k \in C_t$).
```

Algorithm 2: Federated average algorithm.

is, the participants have the same data characteristics but different data samples; it is mainly used in Business-to-Business (B2B) scenarios [10].

2.4.2. Vertical Federated Learning. Vertical federated learning is aimed at data samples with overlapping training data of each client in FL. That is, the data samples between participants are the same, but the data characteristics are different; it is mainly used in Business-to- Client (B2C) scenarios [13].

2.4.3. Federated Transfer Learning. Federated transfer learning is used when the training data characteristics of each participant and the overlap of data samples are relatively small. There are three types of federated transfer learning: case-based, feature-based, and model-based. It is mainly used in retail e-commerce, financial investment, and medical research [10].

Figure 1 shows three categories of federated learning.

The main differences and problems of horizontal federated learning, vertical federated learning, and federated transfer learning are shown in Table 1.

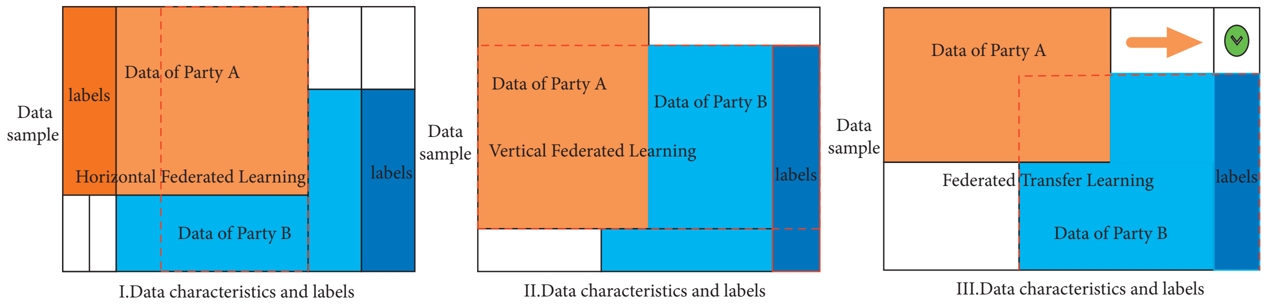
Data sample
labels
Data of Party A
Horizontal Federated Learning
Data of Party B
labels
I.Data characteristics and labels
Data sample
Data of Party A
Data of Party B
Vertical Federated Learning
labels
II.Data characteristics and labels
Data sample
Data of Party A
Federated Transfer Learning
Data of Party B
labels
III.Data characteristics and labels

Figure 1: Three categories of federated learning.

Table 1: Comparison of three kinds of federated learning.

Category	Applicable scenario	Facing the customer	The challenges
Horizontal federated learning	Sample label features are different Sample ID space is the same	B2B	Unable to view distributed training data [10] How to effectively encourage all parties to participate How to prevent the cheating behavior of the participants
Vertical federated learning	Sample label features is the same Sample ID spaces are different	B2C B2B	Establish a reliable and efficient communication mechanism [10] Prevent information disclosure or counterattack
Federated transfer learning	Sample label features are different Sample ID spaces are different	B2B	How to develop a transferable knowledge scheme [10] How to learn the method of transferring knowledge representation How to deploy efficient security protocols in federated migration

3. Unsolved Problems

The definition and classification of FL are described above. This section is mainly on its 5 unsolved problems.

3.1. The Problem of Nonindependent and Identically Distributed Data Samples. In distributed ML, local data samples are often independently and identically distributed. Although FL is a kind of distributed ML, most of its data are nonindependent and uniformly distributed. Moreover, it is different from the batch training in traditional distributed ML; there are some differences in the training data obtained by FL in each round of training. Some scholars tried local data sharing or model migration to solve it, such as federated semisupervised learning and unsupervised learning, which we mention in Chapter 4.

3.2. The Problem That Different Participants Have Different Amounts of Data. The amount of data owned by different clients is different; it is determined by the participants themselves and cannot be controlled. There is a similar problem in Business-to-Business (B2B); some large companies occupy a lot of data resources. How to get such large companies to participate in joint modeling is the first question. The key is to establish a reasonable incentive mechanism to share the profits generated fairly and equitably with participants. Federated blockchain technology can well solve the problem of incentive mechanisms. Paper [13] describes the collection of FL and blockchain technology and how to reward participants.

3.3. Robustness of Participants. In FL, many participants are mobile devices, and different participants have different network structures in data communication. When participating in joint modeling, some methods need to be adopted to ensure the robustness of the model. In addition, there will be some fake participants to attack within the global model established by FL, which we call fake local clients. Some scholars have proposed intrusion detection methods based on federated convolutional neural networks. In the deep learning part, they will specifically introduce how to use deep learning to improve the robustness of the model.

3.4. Communication and Computing Problems. FL means that large-scale data are trained locally, and most of its real application cases are transmitted by wireless communication, so its exchange process requires a stable communication condition. But the task model and data distribution will frequently change with time, and the structure of the federated network, target data characteristics, feature extractors, business tags, etc. will also change, which will lead to communication and computing problems. At present, a large number of papers of this field are proposed, including improving the bandwidth of communication, increasing the stability of transmission, and ensuring the security of communication. In this paper, we will introduce some deep learning algorithms which improved the problem in Chapter 5.

3.5. Privacy and Security of Federated Learning. Although the purpose of FL is to protect the privacy and security of users, in the process of participating in the joint training of FL, even if there is no need to obtain the information of local users, the privacy of users cannot be 100% guaranteed. When constructing a joint model, participating devices need to upload model parameters or gradient values, these parameters come from local models, and the partially trained local devices contain all the information of the data. There are many attacking models and part or all of the original data can be deduced from model parameters or gradients [14, 15], some local device attacks or disguised local model training participants. Therefore, many encryption methods have been proposed, and the common encryption methods are Secure Multiparty Computing (SMC) [16]; Homomorphic Encryption (HE) [17]; Data Disturbance (DD); Differential Privacy (DP) [18];and so on.

In view of the above problems, various researchers have put forward various solutions. In this paper, each method is divided into two categories: one is based on ML and the other is based on pan-deep learning.

4. Federated Learning Algorithm Based on Machine Learning

The most classical Federated Learning Average (FedAvg) is proposed by McMahan et al. [5]; it proves that FedAvg can achieve expected results when tested on the benchmark image classification data set (such as MNIST [19] and CIFAR-10 [20]). Since then, many FL are proposed. Here are several common FL algorithms based on ML; they are classified according to federated supervised learning, federated semi-supervised learning, and federated unsupervised learning. Figure 2 shows the classification of federated ML [21].

4.1. Federated Supervised Learning. Supervised learning is a classic ML method, which infers a functional ML task from marked training data. The training data include a set of training examples. In supervised learning, each instance consists of an input object and the desired output value.

4.1.1. Federated Linear Algorithm. Yang et al. [22] put forward a logical regression method of center-based vertical FL, which realizes logical regression in vertical learning, and the objective function is as follows:

$$\arg\min_{w} L(x, y, w) = \left\{ \frac{1}{N} \sum_{n}^{N} L(w; x_n; y_n) \right\}, \tag{4}$$

where $L(w; x_n; y_n)$ is the loss function, w is the parameter of the model, x_n is the feature of the model, y_n is the label of the model, and $n \in \{1, N\}$ is the amount of data. In the framework of this optimized federated algorithm, homomorphic encryption is added to encrypt the data and gradient of both sides. The whole training process can be described as the data of the unlabeled data holder α are $d_a = w^{\alpha^{\tau}} x$, where $w^{\alpha^{\tau}}$ represents the model parameters of the unlabeled data holder in the τ round state. $[d_\alpha]$ represents the homomorphic encryption of d_α. The unlabeled data holder α first sends $[d_\alpha]$, $[d_\alpha^2]$, and $[\Delta \overline{d}_\alpha]$ to the labeled data holder β, and β calculates the gradient and loss and sends them back after homomorphic encryption. After receiving the encrypted gradients from α and β, the central server assists α and β to update their models.

4.1.2. Federated Support Vector Machine. A federated support vector machine was proposed by Hartmann et al. [23] in 2019. The method optimizes and protects the parameters by updating blocks of local modules, attributing feature hashing and other ways. The objective function is as follows:

$$\arg\min_{w} L(w, x_i, y_i) = \left\{ \frac{1}{N} \sum_{i}^{N} L(w, x_i, y_i) + \lambda R(w) \right\}, \tag{5}$$

where N is the training data, w is the parameters of the model, $L(w, x_i, y_i)$ is the loss at the point (x_i, y_i), $\lambda R(w)$ is the regular term of the loss function, and λ is the hyperparameter to control the penalty. The objective function of support vector machine for traditional ML is as follows:

$$\arg\min_{w} L(w, x_i, y_i) = \{0, 1 - w^{\tau} x_i y_i\}. \tag{6}$$

The Support Vector Machine (SVM) performs dimensionality reduction hash processing on the eigenvalues to hide the actual eigenvalues. The federated support vector machine can update the parameters of the model by updating the gradient of the central server, which can better protect the privacy of the parameters of the model. In practical application cases, the federated support vector machine will not increase calculation, so its actual performance is even better.

4.1.3. Federated Decision Tree Algorithm. Liu et al. [24] proposed a decision tree-oriented vertical federated learning method, a random forest implementation method based on a centralized FL framework, named as Federated Decision Tree (FDT). Its local participants upload the ranking of performance of their model parameters, not model parameters which the original FL constantly uploaded. Thus, it can greatly reduce communication frequency, a large amount of storage, and computing resources consumed by the encryption. In the joint modeling, the model mechanism of the whole random forest is scattered and stored, the central server holds the original complete structural information, and each participating node holds only their own the information [25]. When the federated decision tree model is used, the node information of the local tree is first obtained, and then the other local node information of the tree model is called jointly by

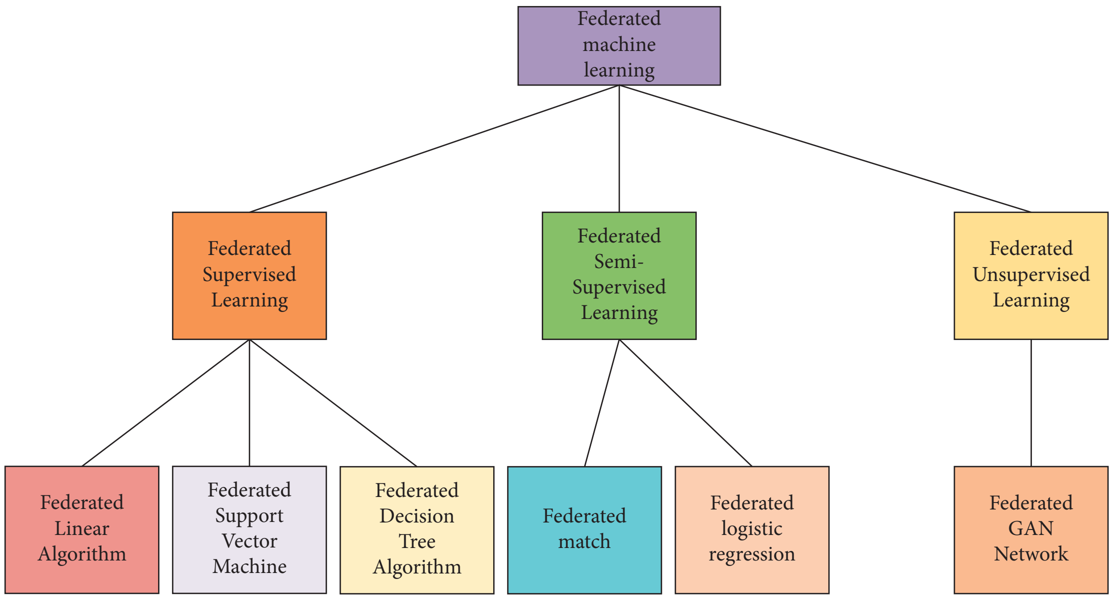
Federated machine learning
Federated Supervised Learning
Federated Semi-Supervised Learning
Federated Unsupervised Learning
Federated Linear Algorithm
Federated Support Vector Machine
Federated Decision Tree Algorithm
Federated match
Federated logistic regression
Federated GAN Network

Figure 2: Federated machine learning classification.

the central server. Among federated decision tree models, Secure Boost model [26] is a decentralized vertical FL framework based on gradient lifting decision tree. According to the common gradient lifting decision tree algorithm, the objective function is as follows:

$$\arg\min_t L^t = \left\{ \sum_n^N j\left(y_n, \hat{y}_i^{(t-1)} + F(x_i) \right) \right\}, \tag{7}$$

where L^t is the minimum loss value of the objective function, t is the tth iteration of the regression tree, $j(y_n, \hat{y}_i^{(t-1)})$ is the loss on the leaf node of each tree function, and $F(x)$ is the sum of the first derivative and the second derivative of the prediction residual. In order to prevent overfitting, a regular term is usually added to the loss function:

$$\phi(f_t) = \gamma L + \frac{1}{2}\lambda w^2, \tag{8}$$

where γ and λ are hyperparameters. In order to adjust the characteristics and the number of trees, w is the weight value and L is the original loss function. In the original distributed ML, joint modeling is realized by sending $F(x)$ to participants, but distributed ML can use $F(x)$ to calculate data labels backward, resulting in data leakage, which does not meet the basic requirements of FL in principle [27]. The federated tree model is based on the Secure Boost [26] encryption algorithm, training the samples of the model that needs joint training, and the first sample and the second sample are trained to get the prediction model of the decision tree. According to the prediction model of the decision tree based on the sample label, it can ensure that the data will not be deduced and calculated in reverse.

Li et al. [28] proposed a decentralized horizontal FL framework for multiparty, named Gradient Boosting Decision Tree (GBDT) modeling—a learning model based on the degree of similarity between data. The encryption degree of hash table encryption is not high, which is not as good as that of differential privacy and federated blockchain [29], but it gives some compensation to communication efficiency when modeling up and down transmission. This is a new research direction of algorithm research under the federated tree model. If data disturbance is added, its confidentiality can be comparable to the differential privacy protection and federated blockchain technology.

4.2. Federated Semisupervised Learning.

Semisupervised learning is a key issue in the field of ML. It can use as much unlabeled data as possible to complete the task [30]. After FL is added to semisupervised learning, on one hand, FL can be used to ensure that sufficient training data are available, and, on the other hand, semisupervised learning can be used to alleviate the problem of the high cost of client-side scattered data labeling.

Jeong et al. [31] proposed a federated semisupervised learning framework according to the number of data tags. Its generative model is mainly to obtain data reconstruction from the perspective of probability, such as $p(x, y) = p(x|y)p(y)$, so it can be estimated by a hybrid model. Recently, VAE [32] and GAN [33] have generated more complex models for semisupervised learning, which further improve the efficiency of semisupervised learning.

According to different split positions of sample identification and feature space, federated semisupervised learning can be divided into two categories: horizontal federated semisupervised learning and vertical federated semisupervised learning [31].

In horizontal federated semisupervised learning, the participating parties $\{1, 2, \ldots, N\}$ have the same feature space χ but different ID logo spatial data of each participant; that is, $I_j \neq I_k, j \neq k$, which is held by all parties involved in the horizontal federated semisupervised learning. For each participant, i has its own data $D_i = D_{iL} \cup D_{iU}$, where

$$D_{iL} = \{X_p \in \chi, y_p\}_{p=1,\ldots,l_i}, \quad l_i \geq 0,$$
$$D_{iu} = \{X_p \in \chi,\}_{p=l_i+1,\ldots l_i+u_i}. \tag{9}$$

Vertical federated semisupervised learning has the same ID logo space of all parties involved, but each party $\{1, 2, \ldots, N\}$ holds different feature space χ; that is, $\chi_j \neq \chi_k, j \neq k$. For each participant, i has its own data $D_i = D_{iL} \cup D_{iU}$, where

$$D_{iL} = \{X_p \in \chi_i, y_p\}_{p=1,\ldots,l_i}, \quad l_i \geq 0,$$
$$D_{iu} = \{X_p \in \chi_i,\}_{p=l_i+1,\ldots,l_i+u_i}. \tag{10}$$

Yang et al. [34] proposed a logical regression method of decentralized longitudinal FL in fact, the label data side to replace the central server. In decentralized vertical FL, data are divided into tagged data and untagged data, in which tagged data are dominant. Assuming that there is an agreement between the unlabeled data holder α and the labeled data holder β to cooperate in modeling, α first sends the modeling key to β, α and β initialize the parameters w^1 and w^2 respectively, and calculate w^{ixi}, where $i \in \{1, 2\}$. After β is calculated, the results are sent to α. α averages both calculation results and then uses logical regression equation to get the final. At last, both tagged and untagged parties are updated by gradient. Table 2 shows the articles of three types of federated machine learning algorithms.

4.3. Federated Unsupervised Learning. Unsupervised learning is a ML method mainly used to discover potential patterns in data. Its input data have no label, and only the input variable (X) is provided, no corresponding output variable (Y).

In unsupervised learning, the algorithm needs to find the pattern structure in the data by itself [35]. The data on each participating client of FL is basically collected in a non-independent and uniformly distributed way, so there is a problem of domain migration between clients. This problem of domain migration makes it difficult to extend the model and its training to new devices. Based on the framework of FL and without user supervision, knowledge is transferred from decentralized nodes to new nodes with different data domains. Peng et al. [36] defined an Unsupervised Federated Domain Adaptive (UFDA) method; it can align the representations learned among the different nodes with the data distribution of the target node. In the domain adaptive system of FL, models on different nodes have different convergence rates. In addition, the domain migration between the source domain and the target domain is different; as a result, some nodes may not contribute to the target domain or even show negative contribution [36].

5. Federated Learning Algorithm Based on Pan-Deep Learning

Federal learning combined with deep learning is one of the mainstreams of federal learning. This chapter focuses on this area; Figure 3 shows a classification of federated pan-deep learning.

5.1. Federated Neural Network. McMahan et al. [37] proposed a federated neural network model and carried out tests on neural networks on MINST data sets. In this paper [37], five groups of experiments are introduced, and this section only introduces the part of neural network (NN). The model has a four-layer network structure, including one input layer, two hidden layers, and an output layer; each hidden layer has 200 neurons. The MINST data set is assigned to each client, and these clients do not intersect. Then the federated training was carried out and the experiment was carried out in two groups: Experiment 1 uses the same random seed to initialize local model parameters allocated to the two clients. Experiment 2 uses different random seeds to initialize local model parameters assigned to the two clients. The different local model parameters of the two groups of experiment are weighted and integrated proportionally to obtain the final federated neural network model, namely,

$$w_{\text{FL}} = \varepsilon w + (1 - \varepsilon)w'. \tag{11}$$

Among them w_{FL} is a federated model parameter, w and w' are model parameters distributed at different nodes, and ε is weight, which changes between 0 and 1. The experiment in this paper shows that when using FL, the federated model with the same random initialization seed has the best effect, and, at the same time, the optimal loss is achieved when the ratio of model parameters is $1:1$.

5.2. Federated Convolutional Neural Network. Zhu et al. [38] proposed a federated CNN; it used a simple CNN network to do text recognition work in unclassified scenarios, and the whole model is built based on TensorFlow and PySyft to test the impact of FL infrastructure and local clients [39]. The built-in reference [38] is a simple CNN with four convolution layers, two fully connected layers, using ReLU activation function, and four output layers defined by the author. The structure of convolutional neural network is described in Figure 4. The CNN classifier is used for dictionary-free text recognition in the Chinese character corpus, and the parameters in CNN are optimized to minimize the aggregate negative logarithmic likelihood of the character sequence:

$$L = \sum_{i=1}^{N} \sum_{k=1}^{4} \sum_{j=1}^{M} -p_{ij}^{(k)} \log \widehat{p}_{ij}^{(k)}, \tag{12}$$

where N is the training data set, M is the total number of classifications, and $p_{ij}^{(k)}$ and $\widehat{p}_{ij}^{(k)}$ are the probability that the kth character of sample i is marked as j. In their experiments, we compared two prevalent federated learning frameworks, namely, TensorFlow Federated and PySyft. Results show that federated text recognition models can achieve similar or even higher accuracy than models trained on deep learning framework. Figure 4 shows the convolutional neural network diagram in [38].

Rong et al. [40] proposed an intrusion detection method based on a federated CNN. This paper uses the data joint training model of multiple participants to expand the number of local participants. Based on the original FL

Table 2: Federated machine learning algorithms.

Classification	Basic algorithm principle	Frame	Paper address
Supervised learning	Logical regression method based on vertical federated learning	Centralization	arXiv: 1912.00513
	Federated learning method based on SVM local module update partition	Centralization	arXiv: 1907.03373
	Random forest longitudinal federated learning method based on decision tree	Centralization	arXiv: 1905.10053
	Vertical federated learning method based on gradient boosting decision tree	Decentralization	arXiv: 1901.08755
	Horizontal federated learning method based on the degree of similarity between data	Decentralization	arXiv: 1911.04206
Semisupervised learning	Federated learning method based on semisupervised learning model	Centralization	arXiv: 2006.12097
	Logistic regression method based on semisupervised longitudinal federated learning	Decentralization	arXiv: 1911.09824
Unsupervised learning	Unsupervised model federated learning method based on GAN network	Centralization	arXiv: 1911.02054

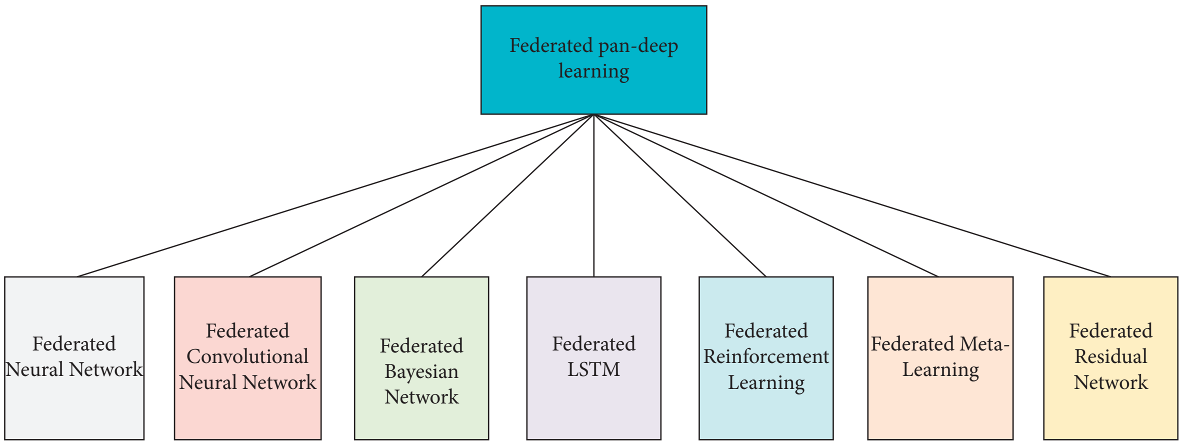
Federated pan-deep learning
Federated Neural Network
Federated Convolutional Neural Network
Federated Bayesian Network
Federated LSTM
Federated Reinforcement Learning
Federated Meta-Learning
Federated Residual Network

Figure 3: Federated deep learning classification.

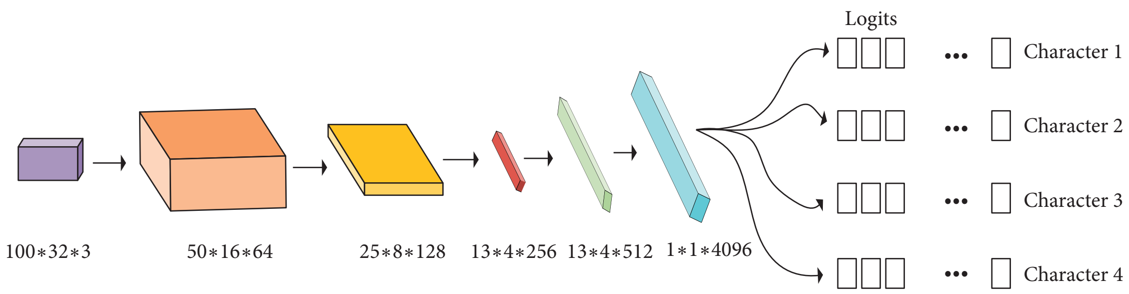

Figure 4: Convolutional neural network.

framework, an intrusion detection model based on deep learning is designed. First of all, the data dimension is reconstructed by data filling to form a two-dimensional array. Then, Diffusion-Convolutional Neural Networks (DCNN) are used to extract and learn the feature parameters under the mechanism of FL. Finally, it is combined with the Softmax classifier training model for detection. This method greatly reduces the training time and maintains a high detection rate. In addition, compared with the general intrusion detection model, the improved model also ensures data security and privacy [40]. Federated convolutional neural networks are generally implemented by a simple CNN model. References [38–40] use a CNN model with four convolution layers and two fully connected layers. This model is suitable for horizontal FL. The ID of the sample is used as the basis, and then the data set is randomly assigned to different clients to form different subsets to simulate distributed data. During the training, the client first carries out gradient calculation and parameter update on the local data set. At the end of each training iteration, the accumulated parameter updates for each client are summarized to update the final federated model.

Three groups of experiments are carried out in paper [40]. In experiment 1, the effectiveness of the method of transforming one-dimensional data into two-dimensional data intrusion detection network is verified. This method not only improves the accuracy of the model but also reduces the operation cost of the model. In experiment 2, the depth of the DCNN model is determined. The experimental results show that the two different models have little change in training and testing time, but in terms of accuracy, the accuracy of the model with two hidden layers is improved by an average of 1%. When it is increased to three hidden layers, the performance is not significantly improved, so simply increasing the number of hidden layers has little effect on the performance improvement. In experiment 3, the intrusion detection model is constructed by the FC algorithm, and the multi-classification experiment is carried out on the NSL-KDD standard data set. The accuracy of the test set has no obvious change; it is optimized in recall rate and false alarm rate, but the optimization effect is obvious in training time. Because the FC model only needs to transmit a small number of parameters when training data, it has certain advantages over other centralized training models in terms of data security. Generally speaking, the federated CNN cannot only improve the security performance in deep learning but also improve the computing power of the model by using GPU.

For the model parameter transfer between clients and the server, in order to reduce the occupation of bandwidth, the CNN is generally compressed. Sattler et al. [41] proposed a new framework of Spare Ternary Compression (STC), which is specially designed to meet FL. The training process of FL includes downloading the model, training the model locally, and updating the trained model to the server for aggregation. The number of bits in which data are transmitted is

$$b^{\text{up/down}} \in \mathcal{O}\left(\underbrace{N_{\text{iter}} \times \varpi}_{\#\text{updates}} \times \underbrace{|\vartheta| \times \left(H\left(\Delta w^{\text{up/down}}\right) + \eta\right)}_{\text{update size}}\right), \tag{13}$$

where N_{iter} is the total number of training iterations performed by each client, ϖ is the communication frequency, $|\vartheta|$ is the size of the model, $H(\Delta w^{\text{up/down}})$ is the entropy of the weight update exchanged during upload and download, and η is the inefficiency of coding, that is, the difference between the real update size and the minimum update size (given by entropy). STC extends the existing top-k gradient sparse compression technology through a new mechanism to achieve downstream compression, internalization, and optimal Golomb coding of weight updates. The existing compression algorithms assume that the local data are independently and identically distributed, and most of the training data in FL are nonindependent and identically distributed data. In the independent and identically distributed data, it is considered that the local gradient is an unbiased estimation of the global gradient; that is,

$$E_{x\sim p_i}\left[\nabla_w l(x,w)\right] = \nabla_w R(w)-, \quad \forall i = 1,2,\ldots,n, \tag{14}$$

where p_i is the data distribution of client i and $R(w)$ is the empirical risk function of the whole data, but this assumption of independent and identically distributed data is difficult to hold in FL, and we can only expect that the mean value of the distribution is unbiased; that is,

$$\frac{1}{n}\sum_{i=1}^{n} E_{x^i \sim p_i}\left[\nabla_w l\left(x^i, w\right)\right] = \nabla_w R(w). \tag{15}$$

The gradient of a single client will be biased towards the local data set:

$$E_{x\sim p_i}\left[\nabla_w l(x,w)\right] = \nabla_w R_i(w) \neq \nabla_w R(w), \quad \forall i = 1,2,\ldots,n. \tag{16}$$

Experiments show that if each edge device sees a unique data distribution, the quality of model training will decline. For neural networks trained with highly skewed non-IID data, the accuracy of FL is significantly reduced by about 55%. It is further proved that the accuracy reduction can be explained by weight divergence and can be quantified by the Empirical Mode Decomposition (EMD) between the distribution of each category and the overall distribution on each device. This paper proposes a strategy: the author improves the training of non-IID data by creating a small part of data that is globally shared among all edge devices. Experiments show that the accuracy of CIFAR-10 data sets containing only 5% globally shared data can be improved by 30%.

5.3. Federated Bayesian Network. Yurochkin et al. [42] proposed to apply Bayesian networks based on FL. Under the assumption that both local data and local models are available, a probabilistic FL framework is developed and studied, with special emphasis on training and aggregation neural network models. Estimated local model parameters (in the case of a neural network, a set of weight vectors) between data sources are matched to build a global network [43, 44]. When the data are available, the method is proposed by training the local model for each data source in parallel. Then, the estimated local model parameters (weight vector group in the case of neural network) are matched between data sources to build a global network. Parameter matching is controlled by the posterior of the Beta-Bernoulli Process (BBP), which is a Bayesian Nonparametric (BNP) model that allows local parameters to match existing global parameters. Or if the existing global parameters do not match, new global parameters are created [42].

The federated Bayesian structure provides several advantages over existing methods [40]. First of all, the federated Bayesian separates the learning of local models from the fusion of local clients to become a global federated model. This decoupling allows us to remain unknown to local learning algorithms, which can be adjusted as needed, and each data source may even use a different learning algorithm. Secondly, given only pretrained models, their

BBP information matching process can combine them into joint global models without additional data or learning algorithms for generating pretrained models. Last but not least, federated Bayesian can effectively learn to compress the federated network from the pretrained local network, and under a moderate communication budget, it can outperform the state-of-the-art algorithm of FL using neural networks. In order to apply the joint probabilistic neural matching method to FL, the feature extractors of Multilayer Perceptron (MLP) sets must be grouped and combined in the process of constructing global feature extractors (neurons). The goal of the Bayesian nonparametric mechanism is to identify the subset of neurons in the J local model that matches the neurons in other local models. Then, the matched neurons are combined to form a global model. Suppose we train J Multilayer Perceptron (MLP) $j = 1, 2, \ldots, J$, and each perceptron has a hidden layer and each sensor has a hidden layer. Let $V_j^{(0)} \in \mathscr{R}^{D\times L_i}$ and $\widetilde{v}_j^{(0)} \in \mathscr{R}^{D\times L_i}$ respectively denote the weight and offset of the hidden layer and $V_j^{(1)} \in \mathscr{R}^{D\times k}$ and $\widetilde{v}_j^{(1)} \in \mathscr{R}^{k}$ represent the weight and offset of the softmax layer. D represents the data dimension, the number of neurons in the hidden layer of BL_i, and k represents the number of classes. We consider a simple architecture:

$$f_j(x) = \operatorname{soft\,max}\left(\sigma\left(xV_j^{(0)} + \widetilde{v}_j^{(0)}\right)V_j^{(1)} + \widetilde{v}_j^{(1)}\right), \tag{17}$$

where $\sigma(\cdot)$ is a nonlinear activation function. A set of weights and deviations $\left\{V_j^{(0)}, \widetilde{v}_j^{(0)}, V_j^{(1)}, \widetilde{v}_j^{(1)}\right\}_{j=1}^{J}$ learns a global neural network with weights and deviations. Figure 5 shows the Bayesian network diagram of a single hidden layer, single-layer probabilistic federated neural matching algorithm. The nodes in the figure represent neurons, and neurons of the same color are matched. This paper uses the corresponding neurons method in the output layer to convert the neurons in each J batch into a weight vector of the reference output layer. Figure 5 shows Bayesian network with hidden layers.

5.4. Federated LSTM. LSTM [45] was proposed in 1997. For its unique design structure, LSTM is suitable for dealing with and predicting important events with long intervals and delays in time series. Some researchers have applied LSTM to the centralization-based FL model to predict the character MINST [46, 47]. LSTM is specially designed to avoid long-term dependency problems. Memorizing long-term information is the default behavior of LSTM in practice [48]. The LSTM is added to the local model training. Its input gate determines the next input parameters, the forget gate loses some parameters, and the output gate outputs the required parameters, which makes the iterative effect better. In LSTM, the first σ on the left is the activation function of the forget gate; the second middle σ and tanh are the activation functions of the input gate; the rightmost σ and the middle tanh are the activation functions of the output gate; x_t is the input, h_t is output, h_{t-1} is the output at the previous time, C_{t-1} is the state at the previous time, and C_t is the state at the current time. Figure 6 shows the internal structure of the LSTM network unit.

The study in [45] proposed to segment the data sets of multiple participating clients. When LSTM is placed in the FL framework, the data are nonindependent and identically distributed, and the appropriate hyperparameters are selected. The nonindependent and identically distributed data model is adjusted to the model accuracy of the conventional situation [46, 47]. Li et al. [49] trained LSTM classifiers in federated data sets and proposed a FL framework federated proximal term (FedProx) to solve statistical heterogeneity for sentiment analysis and character prediction. Compared with traditional FedAvg, FedProx has a faster convergence speed. In the case of system heterogeneity, each local client based on the FedAvg framework cannot complete the variable work according to the change of local client. The FedProx framework proposed in reference [49] introduces a regular term to improve the stability of the whole framework. The essence of the modified term is to increase the limitation of the difference between the parameters in the local model and the parameters in the global model, so as to provide a theoretical basis for explaining the heterogeneity between global and local information. Traditional FedAvg objective function is

$$\arg\min_{w} f(w) = \sum_{k=1}^{N} p_k F_k(w) = E_k, \tag{18}$$

where n_k means that there are n_k samples on the kth device. Generally, it is set to $p_k = n_k/n$, where n is the sum of all n_k, and local functions are minimized F_k. E in FedAvg plays an important role in the convergence of global objective function. The higher the E, the more local computation and less communication between devices, which can effectively improve the overall convergence speed of the global objective function. On the other hand, for the heterogeneous local objective F_k, the E value is too large, which may cause each device to strive to achieve the optimization of its local objective function, rather than the optimization of the global objective function, which will affect the convergence of the global objective function and even lead to divergence. The framework FedProx proposed in this paper [49] is similar to FedAvg in that it selects a subset of devices to participate in the update in each round, performs local updates, and then averages these updates to form global updates. However, FedProx makes some simple and critical modifications to converge. The objective function of ah_k improved FedProx:

$$\arg\min_{w} ah_k\left(w; w^t\right) = F_k(w) + \frac{\mu}{2} w - w^{t2}. \tag{19}$$

A two-layer LSTM classifier with 100 hidden units and 80 embedded layers is used in FedProx.

Its task is to predict the next character, a total of 80 categories of characters. The model takes a sequence of 80 characters as input, embeds each character into a 8-dimensional space, and outputs one character for each training sample after two LSTM layers and a dense connection layer [46]. The experimental results show that FedProx has a faster convergence rate than FedAvg. In particular, in a highly heterogeneous environment, FedProx shows a more stable and accurate convergence behavior than FedAvg, which improves the absolute test accuracy by 22% on average.

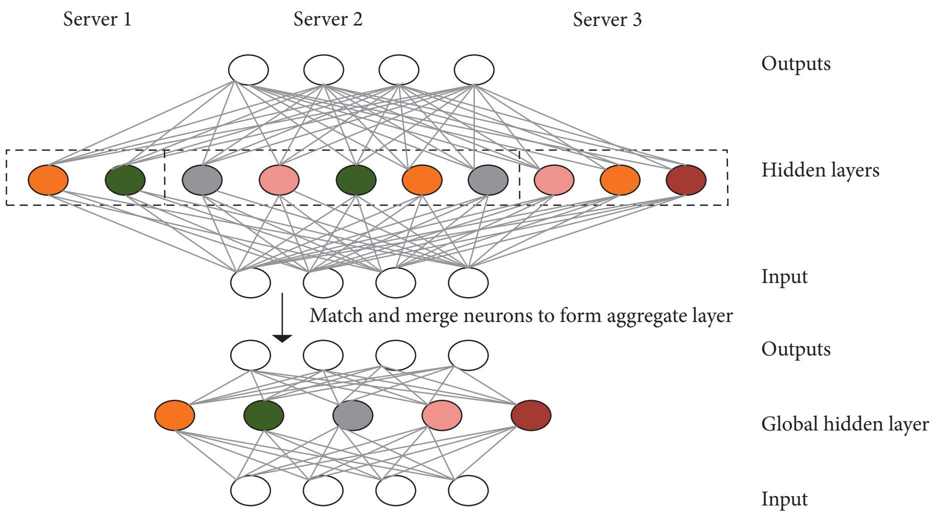

Figure 5: Bayesian network with hidden layers.

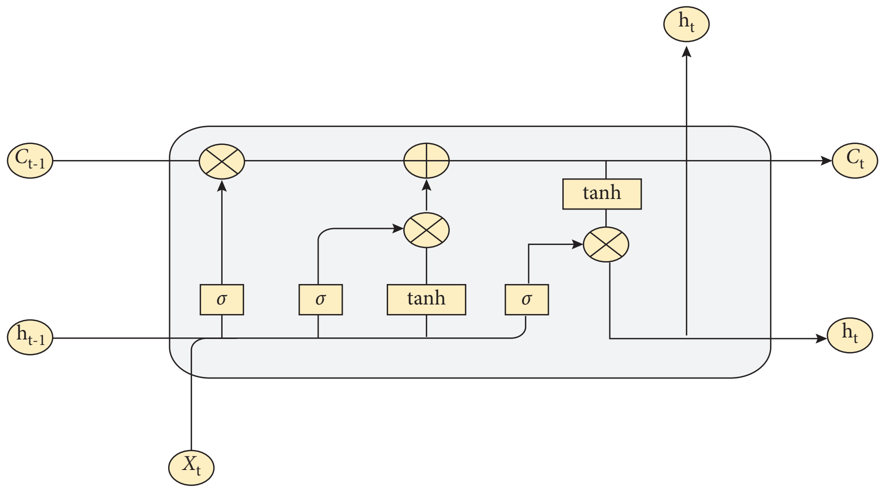

Figure 6: The internal structure of the LSTM network unit.

5.5. Federated Reinforcement Learning. Nadiger et al. [50] first proposed the overall framework of Federated Reinforcement Learning (FRL), which includes grouping strategies, learning strategies, and federated strategies. Reinforcement Learning (RL) and other artificial-intelligence-based technologies have recently been used to achieve personalization. However, reinforcement learning faces the challenge of realizing individualization. In this paper [50], the author proposes a federated reinforcement technology, and its main goal is to improve personalization time. FL, which is applied to reinforcement learning techniques, is an example of hierarchical learning, which enables agents at lower levels to communicate their findings. Local clients with similar environments can be joined more efficiently [51]. The article proposes the use of Deep Q Network reinforcement learning algorithms in a federated environment to achieve faster personalization. The client model and the shared model are regarded as a large Q network and optimized by the Behrman equation. However, in the current work, there is a separate Q-learning on each client, and a joint strategy determines the shared model parameters. The personalized implementation scheme of this article is as follows:

$$\mathrm{PM} = \frac{N_{r4}}{N_r}, \tag{20}$$

where PM refers to a set of games with more personalized measures, N_{r4} is a set of games with long-distance greater than or equal to 4 rounds, and N_r is the total number of gatherings of various lengths in a game. The server sends the global model to all clients. This provides a "hot start" method for each customer. The global model is built offline. Then, the client updates the weight of the Nonplayer Character (NPC) model according to the local RL algorithm. The server starts waiting until the NPC model is received from all customer groups. The global model is

$$w_{t+1}^{g} = \propto w_t^g + (1 - \propto) \sum_{i=1}^{k} \frac{p_i}{p} w_t^c, \quad \text{where, } p = \sum_{i=1}^{k} p_i, \tag{21}$$

where w^g is the global model, w^c is the client model, $\propto$ is the global model regularization factor, the percentage of rebounds with a length greater than or equal to 4 on client i, and k is the number of clients. The experimental results show that this paper proposes a method to speed up the personalization of agents by using federated reinforcement learning. It also puts forward the grouping strategy, learning strategy, and federated strategy, which makes up the whole FRL architecture. The effectiveness of this method is shown by testing on 3, 4, and 5 human players, in which the personalization time is accelerated by about 17%.

Anwar et al. [52] analyzed multitasking federated reinforcement learning from the perspective of confrontation, analyzed the attack performance of many common attack methods, and proposed an adaptive attack method. The general countermeasure is not enough to attack the mobile terminal effectively, so a model poisoning attack method based on minimizing the gain of training information is proposed. In FL, we have multiple local clients. In addition to preventing data poisoning and policy poisoning, we must also consider that the model is attacked. Because we have more than one local client, a complete local client can play the role of an attacker.

Attackers can enter false data and deliberately destroy the federated model. In the attack of federated model, the attacker tries to directly modify the learned model parameters by providing error information that intentionally damages the global model [53, 54]. Because the classical FL uses an average algorithm to merge the local model parameters of a single client learning, such an attack will seriously affect the performance of the global model. In Multitask Federated Reinforcement Learning (MT-FedRL), each client runs in its own environment, which can be characterized by different Markov Decision Processes (MDP). Each agent operates and observes only in its own environment. The goal of MT-FedRL is to learn a unified strategy that is jointly optimal in all n environments. Each agent shares its information with a centralized server. In each of these n environments, the state and action space do not need to be the same. If the state space does not intersect across the environment, the joint problem is decoupled into a set of n independent problems. The goal of the MT-FedRL problem is to find a unified strategy π^* to maximize the sum of the long-term discounted returns for all environments, namely,

$$\arg\min_{\pi} V(\pi;\rho) = \sum_{i=0}^{n-1} \mathbb{E}_{s_i \sim \rho_i} V_i^{\pi}(s_i), \quad \rho = \begin{bmatrix} \rho_0 \\ \vdots \\ \rho_{n-1} \end{bmatrix}. \tag{22}$$

Solving the abovementioned equations will produce a uniform π^*, thus achieving balanced performance in all environments. Where V_i^{π} is the value function of the strategy π^*, in the states of the ith environment, we use ρ to represent the initial state distribution on the action space of the ith environment. In this article [55], it is proved that multitask federated reinforcement learning can converge to a unified strategy, which can achieve the best performance in every environment. If the client's goals are positively correlated, this joint optimal strategy works best when evaluated in each environment. If the client's goals are not positively correlated, a unified strategy may not produce a near-optimal strategy for a single environment. In this article, three common attack models are discussed in detail: the random strategy attack model, the reverse target strategy attack model, and the counterattack model with minimum information gain. Finally, we propose a modification of the general federated reinforcement learning algorithm to solve the antiattack problem, which is equally effective with and without attacks. The federated reinforcement learning process and federated reinforcement learning algorithm are given in reference [52], in which several cooperative models try to maximize the sum of discounted returns in the presence of hostile models in different environments. Figure 7 shows the flow chart of federated reinforcement learning.

5.6. Federated Meta-Learning. Chen et al. [56] proposed a Federated Meta-learning (FedMeta) framework, which shares parameters rather than the previous global model. This article evaluates the LEAF data set and the actual data set and proves that the communication cost required by FedMeta is reduced by 2.82–4.33 times, and its convergence speed is faster, compared with FedAvg, by 3.23%~14.84%. In the field of FL, the local model uses SGD training to achieve high accuracy while balancing the computational and communication costs; in the field of meta-learning, the MAML algorithm is used to quickly converge on new tasks and show good generalization; on this basis, a federated meta-learning framework was built. The FedMeta framework integrates the MAML algorithm and meta-sgd into FL, which improves the accuracy of the joint training model and reduces the communication overhead. Meta-learning A_{φ} algorithm is where φ uses a set of task updates in the meta-training process, and the task test in the meta-training consists of a support set $D_S^T = \{(x_i, y_i)\}_{i=1}^{|D_S^T|}$ and query set $D_Q^T = \{(x_i', y_i')\}_{i=1}^{|D_Q^T|}$ both containing marked data points [57]. Algorithm A trains a model f on the support set D_S^T and outputs D_Q^T called internal update, evaluates the model f_{w_T} on the query set D_Q^T, and calculates the test loss $L_{D_Q^T}(w_T)$ to reflect the training ability of A_{φ} [58]. Finally, A_{φ} is updated to minimize test loss, which is called external update. Each episode of meta-learning algorithm A will sample a batch of tasks from a meta-training set, so the optimization goal of meta-train can be expressed as

$$\arg\min_{\phi} \mathbb{E}_{T\sim\tau}\left[L_{D_Q^T}(w_T)\right] = \arg\min_{\phi} \mathbb{E}_{T\sim\tau}\left[L_{D_Q^T}\left(A_{\varphi}\left(D_S^T\right)\right)\right]. \tag{23}$$

For each task T, the algorithm makes $\varphi = w$, so that the parameters of the algorithm are equal to those of the model f. Then the parameters of model f are trained on the support set and updated according to the loss function:

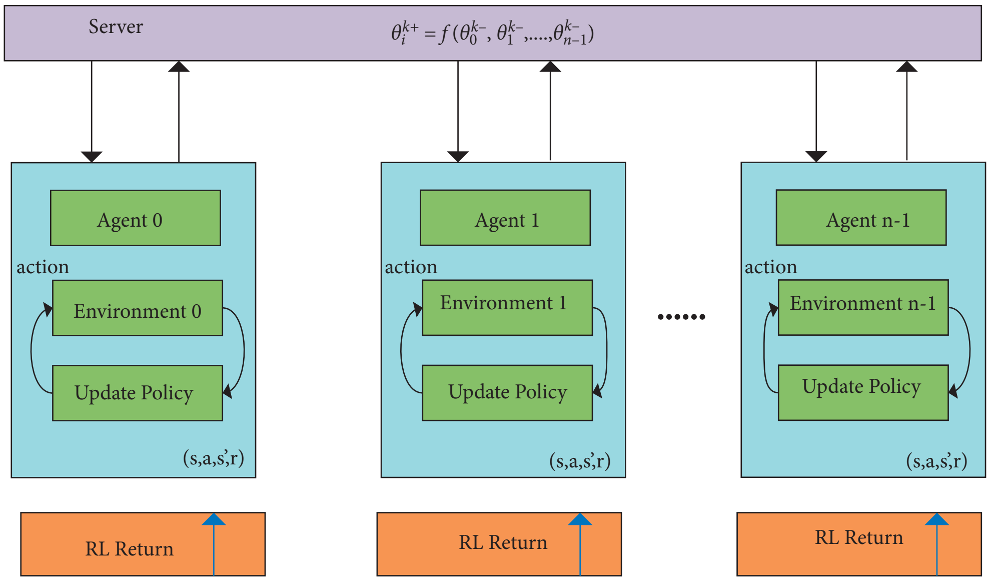

Figure 7: Flow chart of federated reinforcement learning.

$$L_{D_Q^T}(w_T) = \frac{1}{\left|D_S^T\right|} \sum_{(x,y)\in D_S^T} \ell(f_w(x), y). \tag{24}$$

Finally, the model parameters are tested on query set, and then the loss function of the test is calculated:

$$L_{D_Q^T}(w_T) = \frac{1}{\left|D_S^T\right|} \sum_{(x',y')\in D_S^T} \ell(f_w(x'), y'). \tag{25}$$

The experiment is verified on the LEAF data set, which shows that the convergence speed is faster and the accuracy is greatly improved over the traditional FL. At the same time, it also reduces the cost of communication. The goal of meta-learning is to train an algorithm. Federated meta-learning means that many devices join together to train the same meta-learner. Each device has its own meta-learner, but the parameters are aggregated on the server, and then the global meta-learner is trained. The global model trained by FL is the same on every device. Because of the strong data heterogeneity of each device, it is necessary to use meta-learning to personalize the model. Meta-learning generates a metamodel locally, and then metamodels generate personalized models locally, which are suitable for local heterogeneous data. Figure 8 shows the federated meta-learning framework.

5.7. Federated Residual Network. Huang et al. [59] proposed a new compression strategy Residual Pooling Network (RPN) [60] in order to improve the communication efficiency of FL. Compared with traditional FL, RPN alleviates the problem of communication computing overhead by selecting appropriate parameters and can maintain the original performance while reducing data transmission. RPN is an end-to-end process, and it can also be applied to CNN-based model training scenarios to improve the communication efficiency of federated models. The total number of bits that must be transmitted during model training is given by

$$S_{\text{total}} = \sum_{t=1}^{T} \left\{ F\left(G^t\right) + \sum_{i=1}^{M} F\left(G_i^t\right) \right\}, \tag{26}$$

where T is the total number of iterations, M is the number of clients that the server chooses to update in the T round, G^t represents the global model after t aggregations, and $F(G^t)$ is downloaded to the optional parameter bit of the client. Similarly, $F(G_i^t)$ is the selected parameter bit of the client i used for uploading to the server. The article improves communication efficiency from four aspects: iterative frequency, pruning, importance-based update, and quantification. R_i^t is defined as a residual network, and the definition of a residual network is given in the following formula:

$$R_i^t = G_i^t - G_i^{t-1}, \tag{27}$$

where G_i^t is the i parameterized by ; that is, $G^t = f(w^t)$. The experiment in this article includes classification, object detection, and semantic segmentation. They prove that RPN not only effectively reduces data transmission but also achieves almost the same performance as traditional FL. Most importantly, RPN is an end-to-end process, which makes it easy to deploy in real-world applications without human intervention. The federated residual network learning workflow includes (1) selecting clients for local model updates, (2) restoring local models, (3) training local model based on local data sets, (4) calculating remaining networks, (5) spatial aggregation, (6) sending RPN to the server and aggregating, and (7) sending RPN back to the selected client and repeating the cycle. Figure 9 is a schematic diagram of the federated residual network.

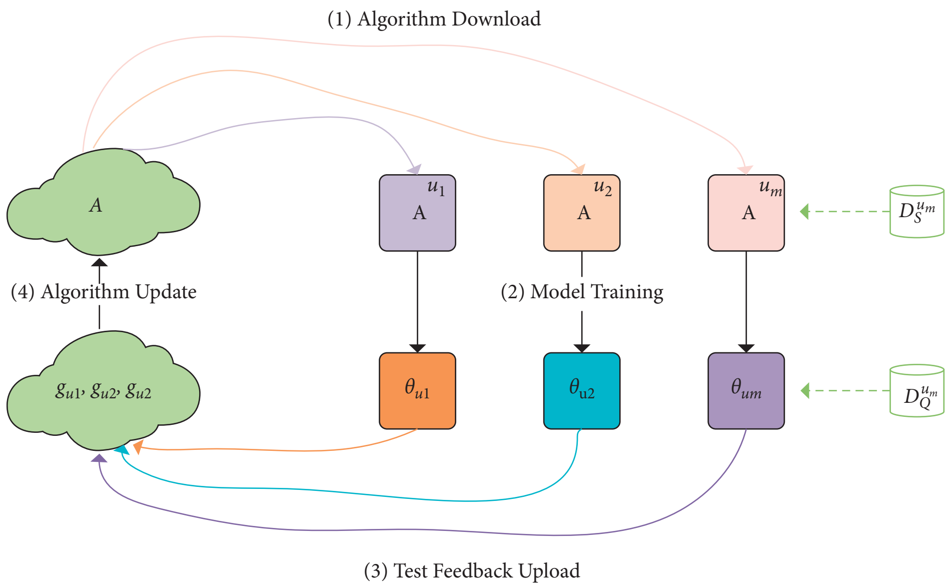

Figure 8: Work flow chart of the federated meta-learning framework.

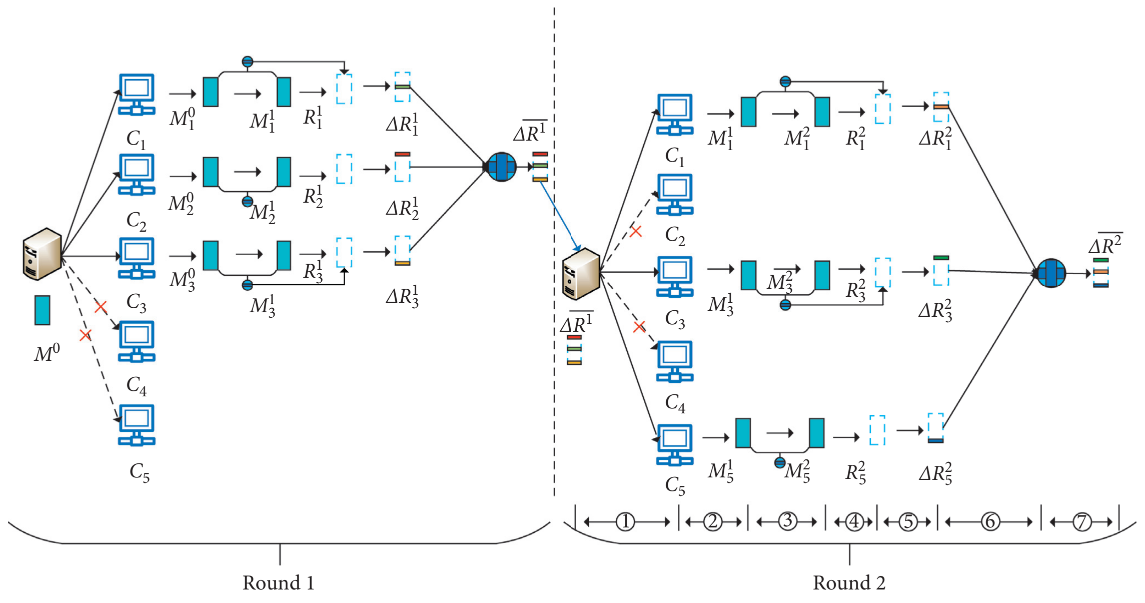

Figure 9: Federated residual network workflow.

Table 3 shows the current federated learning methods based on deep learning.

6. Privacy and Security Issues of Federated Learning

Although FL can ensure that the data are trained locally on the client, it still has privacy and security issues in the event of malicious attacks, which are mainly reflected in the following three aspects. Firstly, the data collector collects user data privately without permission, leading to direct data leakage during data collection; secondly, there is indirect privacy leakage due to insufficient generalization ability of the model; finally, the model may be polluted for the lack of safety precaution [61]. This section discusses the prevention and attack aspects of FL.

6.1. Byzantine Prevention of Federated Learning. In recent years, security issues in FL have attracted widespread attention; especially in some scattered environments, some unstable clients may behave abnormally and even have Byzantine failures—arbitrary and potentially hostile behaviors [62]. Byzantine-robust FL aims to accurately learn the global model on the server side when a limited number of clients are malicious. The key idea of the existing Byzantine-robust FL is that the service provider performs statistics in the client's local model update and removes the available models before aggregating them to update the global model

Table 3: Federated deep learning algorithm.

Classification	Basic algorithm principle	Frame	Paper address
NN	Federated learning method based on traditional neural network	Centralization	arXiv:1610.02527
CNN	Federated learning method based on convolutional neural network	Centralization	arXiv:1902.01046
Bayesian network	Federated learning method based on Bayesian network	Centralization	arXiv:1905.12022
LSTM	Federated learning method based on long-term and short-term memory neural network	Centralization	arXiv:1610.02527 arXiv:1812.06127
RL	Federated learning method based on reinforcement learning	Centralization	arXiv: 2103.06473
ML	Federated learning method based on meta-learning	Centralization	arXiv: 1802.07876
RN	Federated learning method based on residual network	Centralization	arXiv: 2001.08600

[63]. At present, the main vulnerability of FL is the concern of SGD. How to ensure the robustness of distributed SGD and sending poisonous hostile Byzantine clients in the training phase is a hot research topic [64]. In the learning process of the hostile Byzantine client, the learning model may be biased due to data corruption, communication failure, or malicious sending of incorrect information to the server side [65]. Learning the defense against the Byzantine problem, Blanchard [53] proposed Krum, the first provable Byzantine algorithm for distributed SGD, which satisfies the elasticity of the aggregation rule. In face of potential abnormal clients, Yin et al. [62] proposed two robust distributed gradient descent algorithms based on median and pruning average operations for sharp analysis and proved that the distributed algorithm based on median is robust, having the same optimal fault tolerance rate of the distributed gradient descent algorithm. Li et al. [65] proposed the Byzantine-Robust Stochastic Aggregation (RSA) method. RSA regularizes the objective function to enhance the robustness of the learning task. Compared with most algorithms, the RSA method can adapt to independent and identically distributed FL, so it is suitable for a wider range of applications. Shejwalkar and Houmansadr [66] proposed divide-and-conquer (DnC) and demonstrated that DnC outperforms all existing Byzantine-robust FL algorithms in defeating model poisoning attacks.

6.2. Local Model Attack of Federated Learning. In addition, some researchers study the robustness of FL from the attack method. The attacks of FL mainly come from internal attackers participating in the FL process and unique model training strategies. Malicious opponents may interfere with or backdoor the process of distributed learning. Baruch et al. [67] proposed a new attack method, through limited changes to many parameters, in Moran's paper, a variant of trimmed mean is to be chosen among existing defenses, producing the best results for convergence attack excluding the choice of naive averaging, which is obviously vulnerable to other simpler attacks [67].

Bhagoji et al. [68] explored the threat of model poisoning attacks on federated learning, initiated by a single, non-colluding malicious agent where the adversarial objective is to cause the model to misclassify a set of chosen inputs with high confidence. They use a suite of interpretability techniques to generate visual explanations of model decisions for both benign and malicious models and show that the explanations are nearly visually indistinguishable. Their results indicate that even a highly constrained adversary can carry out model poisoning attacks while simultaneously maintaining stealth, thus highlighting the vulnerability of the FL setting, to develop effective defense strategies [68]. Bagdasaryan et al. [69] used the privacy protection mechanism of FL and added abnormal data to carry out vicious attacks on the model, making the existing Byzantine anomaly detection unrecognizable. So how to design robust FL systems is an important topic for future research. Fang et al. [70] performed the first systematic study on local model poisoning attacks to FL. They assume an attacker has compromised some client devices, and the attacker manipulates the local model parameters on the compromised client devices during the learning process such that the global model has a large testing error rate. Experiments show it is valuable future work to design new defenses against local model poisoning attacks, new methods to detect compromised local models, and new adversely robust aggregation rules [70].

7. Future Challenges

7.1. Data Privacy Issues. Under the framework of FL, although the user's local data do not need to be uploaded to the server, it will be directly used in local modeling. If you do not independently add noise to these local data to protect their security, an attack by a malicious user may take place [71]. There are two modes of attack: one is an active attack, and the other is a passive attack.

When setting the FL algorithm protocol, if we assume that the active participant is a malicious attack, which destroys the security performance of the model, we call the malicious attack of the active participant the active attacker of FL. The server can obtain the model update parameters from various devices, and it can carry out FL model attacks by analyzing the model parameters of each round of updates.

We call semihonest but curious server-side attacks passive attackers. The main difference between the active attacker and the passive attacker is that the attack behavior is initiated by different malicious users, the initiating user

Table 4: Federated learning problems.

Classification	Solutions	Paper address
Data privacy issues	Add noise to data	arXiv:2006.11601
	Model encryption	arXiv:2010.10996
Data communication issues	Communication optimization	arXiv:2005.05238
Data heterogeneity issues	Individuation	arXiv:2103.00710
Data overhead issues	Distributed computing	arXiv:2007.13518
Data overhead issuer	Establish a trusted server	arXiv:1902.01046

of the active attacker is the client, and the initiating user of the passive attacker is the server. Both types of attacks damage the confidentiality, integrity, and availability of the FL model [72, 73]. The attacked federated model and the jointly trained model will lose their balance. In the worst case, the jointly established model cannot be returned to the local client.

7.2. Data Communication Issues. In the framework of FL, client-side and server-side devices communicate and transmit model parameters or gradients, and its communication rate is more frequent than the traditional distributed machine transmission rate. But each model participating in joint training cannot have the same computing power and stable transmission rate, which will often cause communication instability. For example, the input method of mobile phone uses FL, some mobile phones use mobile data and some mobile phones carry out joint modeling in WIFI state, the stability of data transmission in mobile data state is usually worse than that in WIFI state, and it is easy to cause communication interruption when uploading or downloading model parameters. Even if the same mobile phone is in the same network state, the communication will be unstable due to the different number of parameters transmitted. Therefore, in the modeling of FL, the data communication problem is a problem worthy of various researchers to ponder. In addition, the problems of communication bandwidth proposed in [74], the convergence of the joint training model, and the communication between cloud service providers are all problems that need to be researched.

7.3. Data Heterogeneity Issues. The data of distributed ML are often independent and identically distributed, but FL is different from traditional distributed ML. Devices in FL often exist in the network in a nonindependent and uniformly distributed way. The data participating in training is generally nonindependent and identically distributed. For example, banks and Internet shopping, although they have the same customers to some extent, their data storage structures are heterogeneous. In addition, the uneven distribution of data held by cross-device data holders will also lead to data heterogeneity. Therefore, many common algorithms for independent and identically distributed data cannot be used directly. How to research algorithms that are more compatible with FL heterogeneous data is also a very important development direction of FL.

7.4. Data Overhead Issues. In the application scenario of FL, most of the local models that participate in the training need to perform computing and communication tasks on mobile terminal. Because the number of local models involved is very large, it is not only a challenge for the communication but also a great test for computing. FL is not only technical labeling but also a business model. Encryption is a very important link in the financial industry, and the original cloud computing model has been challenged in encryption. Adding the encryption algorithm to cloud computing data transmission is a common encryption method. Some researchers have proposed secure mixing [74] and secure mixing [75] methods, but this increases the cost of communication. After adding the encryption method, it needs to be decrypted, and the computational cost of the model data is further increased. Literature [76] puts forward the problem of keeping a balance between communication cost and accuracy and guides the balance between them by evaluating the distributed statistics and learning rate of a certain bandwidth. At present, no researchers have applied it to FL, and there is no up-to-date method to solve the problem of high data computing overhead, so this field needs people to open up and improve. The problem of data computing overhead is urgently waiting to be solved.

7.5. Lack of a Trusted Central Server. In the process of FL, a trusted central server is needed to ensure the privacy and security of users. Some scholars put forward the decentralized algorithm, which is based on the local update scheme of heterogeneous data decentralization training. FL requires a central server to coordinate the training process and receive models uploaded by all clients. Therefore, the server is a central participant, and it may also have a single point of failure. Although large companies or organizations can play this role in certain application scenarios, in more collaborative learning scenarios, a reliable and powerful central server may not always be available. Even if centralized differential privacy is adopted in the protection of data, the central server must be trusted by users. Otherwise, it will cause data leakage. Future researchers can start with how to build a trusted central server to further improve the server structure of FL, making it less vulnerable to attacks and failure. The existing trusted server transformation mainly includes ARM's Trust-Zone architecture and Intel's SGX-enabled CPU architecture [59]. Table 4 shows the federated learning problem.

Table 5: Description of common symbols.

α	The unlabeled data holder in the client
β	The labeled data holder in the client
δ	Loss of accuracy between predicted and real values
$\propto$	Global model regularization factor in the federated reinforcement learning
$\sigma(\cdot)$	Nonlinear activation function
$\lvert\vartheta\rvert$	The model size of the spare ternary compression
λ	Super-parameter control punishment of the model
φ	A set of task updates in the meta-training process
τ	The τ iteration in the federated learning
γ	Super-parameter in the federated learning
μ	Model iteration step of federated learning
η	Inefficiency of coding in federated learning CNN
χ	Feature space of data set
ϖ	The communication frequency
$\sum$	Summation symbol
ah_k	Objective function in FedProx
a_i	The action space in federated reinforce learning
BL_i	The number of neurons in hidden layer of Bayesian network
$b^{\mathrm{up/down}}$	Number of bits to transmit data in federated learning
C_t	The state at the current time
C_{t-1}	The state at the previous time
d_α	Untagged data holder α data
$E_{x\sim p_i}[\nabla_w l(x,w)]$	The local gradient is an unbiased estimation of the global gradient
F	The corresponding rule
$F(G^t)$	Optional parameter bits downloaded to the client in federated residual network
$F(G_i^t)$	Selected parameter bits for client i uploaded to the server in federated residual network
$\{F_1, F_2, \ldots, F_k\}$	The scenario for federated learning is decentralized multiuser
h_t	t time output of the output gate
h_{t-1}	$t-1$ time output of the output gate
$H(\Delta w^{\mathrm{up/down}})$	Entropy of weight updates exchanged during upload and download
k	The client number
L	Loss function
$L(w, x_i, y_i)$	The loss function at the point (x_i, y_i)
M_{SUM}	Traditional distributed machine learning aggregation model
M_{FED}	Average model of federated learning
n_k	There are n_k samples on the k_{th} device
p_k	Weight value of the k th client
V_{FED}	Defined in the performance of M_{FED} on the test set
V_{SUM}	Defined in the performance of M_{SUM} on the test set
r	The learning rate
softmax	Normalized function
S_{total}	The total number of digits that must be transmitted during model training
$\tanh(\cdot)$	Activation function
w	Parameters of the model
x	The client input data
Y	The client output data
x_n	Characteristics of the model in federated learning
y_n	Label of the model in federated learning

8. Conclusion

This paper discusses the classification and development of FL and several existing problems of FL. It expounds from the point of view of FL algorithm, focusing on federated deep learning on the basis of the introduction of federated ML. In the chapter of federated deep learning, existing deep learning algorithms are discussed from the perspectives of communication, data heterogeneity, privacy protection, and trusted server in FL. At present, FL is still in the stage of rapid development, and there are still many unsolved problems about ML and deep learning algorithms under the framework of FL. With the further expansion of the amount of data in the future, the implementation of deep learning algorithm is not only a feasible scheme for practicing in the field of artificial intelligence but also a more efficient and comprehensive method for the use of distributed ML and edge data. In the future, FL will develop incoordination in multiple fields, such as edge computing, blockchain, privacy protection, and other coordinated development to improve the performance of FL and, at the same time, make the commercial value better. In order to facilitate readers to understand common symbols in this paper, we have added a symbol table, shown in Table 5.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

All authors drafted, read, and approved the final manuscript.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (42075130, 61773219, and 61701244) and the Key Special Project of the National Key R&D Program (2018YFC1405703).

References

[1] C. Tikkinen-Piri, A. Rohunen, and J. Markkula, "EU general data protection regulation: changes and implications for personal data collecting companies," *Computer Law & Security Report*, vol. 34, no. 1, pp. 134–153, 2018.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[3] J. X. Liu and X. F. Meng, "Survey on privacy preserving machine learning," *Journal of Computer Research and Development*, vol. 57, no. 2, pp. 346–362, 2020, in Chinese.

[4] X. G. Li and F. H. Li, "A survey on differ entail privacy," *Journal of Cyber Security*, vol. 3, no. 5, pp. 92–104, 2018, in Chinese.

[5] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," pp. 1273–1282, 2017, https://arxiv.org/abs/1602.05629.

[6] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," 2016, https://arxiv.org/abs/1610.05492.

[7] G. Yang and Z. S. Wang, "Survey on privacy preservation in federated learning," *Journal of Nanjing University of Posts and Telecommunications (Natural Science Edition)*, vol. 40, no. 5, pp. 204–221, 2020.

[8] Z. Chen and B. Liu, "Lifelong machine learning, second edition," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.

[9] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1310–1321, Monticello, IL, USA, September 2015.

[10] Q. Yang and Y. Liu, "Federated learning: the last on kilometer of artificial intelligence," *Journal of Intelligent Systems*, vol. 15, no. 1, pp. 183–186, 2020.

[11] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[12] P. Kairouz, H. B. McMahan, B. Avent et al., "Advances and open problems in federated learning," 2019, https://arxiv.org/abs/1912.04977.

[13] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.

[14] R. Pathak and M. J. Wainwright, "Fed split: an algorithmic framework for fast federated optimization," 2020, https://arxiv.org/abs/2005.05238.

[15] L. Zhu and S. Han, "Deep leakage from gradients," *Lecture Notes in Computer Science-Federated Learning*, Springer, Berlin, Germay, pp. 17–31, 2020.

[16] N. Kilbertus, A. Gascón, M. Kusner, M. Veale, K. Gummadi, and A. Weller, "Blind justice: fairness with encrypted sensitive attributes," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 2630–2639, Stockholm, Sweden, July 2018.

[17] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on Machine Learning*, pp. 201–210, New York, NY, USA, June 2016.

[18] C. Dwork, "Differential privacy: a survey of results," in *Proceedings of the International Conference on Theory and Applications of Models of Computation*, pp. 1–19, Springer, Xi'an, China, April 2008.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of Systemic Autoimmune Diseases*, vol. 1, no. 4, 2009.

[21] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: reducing the communication bandwidth for distributed training," 2017, https://arxiv.org/abs/1712.01887.

[22] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[23] V. Hartmann, K. Modi, J. M. Pujol, and R. West, "Privacy-preserving classification with secret vector machines," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 475–484, New York, NY, USA, October 2020.

[24] Y. Liu, Y. Liu, Z. Liu et al., "Federated forest," *IEEE Transactions on Big Data*, 2020.

[25] J. Z. Wang, L. W. Kong, Z. Huang et al., "Research review of federated learning algorithms," *Big Data Reserch*, vol. 6, no. 6, pp. 70–88, 2020.

[26] K. Cheng, T. Fan, Y. Jin et al., "Secure boost: a lossless federated learning framework," 2019, https://arxiv.org/abs/1901.08755.

[27] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving distributed federated learning with federated learning," 2021, https://arxiv.org/abs/2004.12108.

[28] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 4642–4649, 2020.

[29] H. Kim, J. Park, M. Bennis, and S. L. Kim, "Block chained on-device federated learning," 2019, https://www.arxiv-vanity.com/papers/1808.03949/.

[30] E. M. Tu and J. Yang, "A review of semi-supervised learning theories and recent advances," *Journal of Shanghai Jiaotong University*, vol. 52, no. 10, pp. 1280–1291, 2018.

[31] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, "Federated semi-supervised learning with inter-client consistency," 2020, https://arxiv.org/abs/2006.12097.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, https://arxiv.org/abs/1312.6114.

[33] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semisupervised learning with deep generative models," 2014, https://arxiv.org/abs/1406.5298.

[34] S. Yang, B. Ren, X. Zhou, and L. Liu, "Parallel distributed logistic regression for vertical federated learning without third-party coordinator," 2019, https://arxiv.org/abs/1911.09824.

[35] S. Peng, *Research on Anomaly Detection Technology in Surveillance Video Based on Unsupervised Learning*, Beijing University of Technology, Beijing, China, 2019.

[36] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," 2019, https://arxiv.org/abs/1911.02054.

[37] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, Fort Lauderdale, FL, USA, 2017.

[38] X. Zhu, J. Wang, Z. Hong, T. Xia, and J. Xiao, "Federated learning of unsegmented Chinese text recognition model," in *Proceedings of the 2019 IEEE 31st international Conference on tools with artificial intelligence (ICTAI)*, pp. 1341–1345, IEEE, Portland, OR, USA, November 2019.

[39] M. Xia, X. Zhang, W. A. Liu, L. Weng, and Y. Xu, "Multi-stage feature Constraints learning for age estimation," *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 1, pp. 2417–2428, 2020.

[40] W. Rong, M. A. Chunguang, and P. Wu, "An intrusion detection method based on federated learning and convolutional neural network," *Netinfo Security*, vol. 20, no. 4, p. 47, 2020.

[41] F. Sattler, S. Wiedemann, K. R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[42] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proceedings of the International Conference on Machine Learning*, pp. 7252–7261, Long Beach, CA, USA, 2019.

[43] B. Chen, M. Xia, and J. Huang, "MFANet: a multi-level feature aggregation network for semantic segmentation of land Cover," *Remote Sensing*, vol. 13, no. 4, p. 731, 2021.

[44] M. Xia, Y. Cui, Y. Zhang, Y. Xu, J. Liu, and Y. Xu, "DAU-net: a novel water areas segmentation structure for remote sensing image," *International Journal of Remote Sensing*, vol. 42, no. 7, pp. 2594–2621, 2021.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: building an efficient and scalable deep learning training system," in *Proceedings of the 11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pp. 571–582, Broomfield, CO, USA, October 2014.

[47] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: distributed machine learning for on-device intelligence," 2016, https://arxiv.org/abs/1610.02527.

[48] M. Xia, W. a. Liu, K. Wang, W. Song, C. Chen, and Y. Li, "Non-intrusive load disaggregation based on composite deep long short-term memory network," *Expert Systems with Applications*, vol. 160, Article ID 113669, 2020.

[49] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, https://arxiv.org/abs/1812.06127.

[50] C. Nadiger, A. Kumar, and S. Abdelhak, "Federated reinforcement learning for fast personalization," in *Proceedings of the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 123–127, IEEE, Sardinia, Italy, June 2019.

[51] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," pp. 1273–1282, 2017, https://arxiv.org/abs/1602.05629.

[52] A. Anwar and A. Raychowdhury, "Multi-task federated reinforcement learning with adversaries," 2021, https://arxiv.org/abs/2103.06473.

[53] P. Blanchard, E. M. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 118–128, Long Beach, CA, USA, December 2017.

[54] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 634–643, Long Beach, CA, USA, 2019.

[55] S. Zeng, A. Anwar, T. Doan, A. Raychowdhury, and J. Romberg, "A decentralized policy gradient approach to multi-task reinforcement learning," 2020, https://arxiv.org/abs/2006.04338.

[56] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated Meta-learning with fast convergence and efficient communication," 2018, https://arxiv.org/abs/1802.07876.

[57] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, pp. 1126–1135, Sydney, Australia, August 2017.

[58] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: learning to learn quickly for few-shot learning," 2017, https://arxiv.org/abs/1707.09835.

[59] A. Huang, Y. Chen, Y. Liu, T. Chen, and Q. Yang, "RPN: a residual pooling network for efficient federated learning," 2020, https://arxiv.org/abs/2001.08600.

[60] M. Xia, T. Wang, Y. Zhang, J. Liu, and Y. Xu, "Cloud/shadow segmentation based on global attention feature fusion residual network for remote sensing imagery," *International Journal of Remote Sensing*, vol. 42, no. 6, pp. 2022–2045, 2021.

[61] J. Shi and Y. Zhou, "Study on privacy protection techniques of federated leraning," *Modern Information Technology*, vol. 5, no. 1, pp. 138–142, 2021.

[62] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, July 2018.

[63] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: byzantine-robust federated learning via trust bootstrapping," 2020, https://arxiv.org/abs/2012.13995.

[64] R. Guerraoui and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," https://arxiv.org/abs/1802.07927.

[65] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 1544–1551, 2019.

[66] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning," *Internet Society*, p. 18, 2021.

[67] M. Baruch, G. Baruch, and Y. Goldberg, "A little is enough: circumventing defenses for distributed learning," 2019, https://arxiv.org/abs/1902.06156.

[68] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," 2019, https://arxiv.org/abs/1811.12470.

[69] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," 2020, https://arxiv.org/abs/1807.00459.

[70] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," 2019, https://arxiv.org/abs/1911.11815.

[71] S. Truex and N. Baracaldo, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pp. 1–11, London, UK, November 2019.

[72] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: a client level perspective," 2017, https://arxiv.org/abs/1712.07557.

[73] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[74] K. Bonawitz, H. Eichner, W. Grieskamp et al., "Towards federated learning at scale: system design," 2019, https://arxiv.org/abs/1902.01046.

[75] A. Bittau, Ú. Erlingsson, P. Maniatis et al., "Prochlo: strong privacy for analytics in the crowd," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 441–459, Shanghai, China, October 2017.

[76] S. Sasy, S. Gorbunov, and C. W. Fletcher, "ZeroTrace: oblivious memory primitives from intel SGX//NDSS," in *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium 2018*, San Diego, CA, USA, February 2018.