Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8967819, 14 pages
https://doi.org/10.1155/2021/8967819

Research Article

Federated Learning Model with Adaptive Differential Privacy Protection in Medical IoT

Lina Ni,[1,2] **Peng Huang**,[1] **Yongshan Wei**,[1] **Minglei Shu**,[3] **and Jinquan Zhang**[1,2]

[1]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China*
[2]*Key Laboratory of the Ministry of Education for Embedded System and Service Computing, Tongji University, Shanghai 201804, China*
[3]*Qilu University of Technology (Shandong Academy of Sciences), Shandong Artificial Intelligence Institute, Jinan 250353, China*

Correspondence should be addressed to Jinquan Zhang; tjzhangjinquan@126.com

Received 21 May 2021; Revised 10 October 2021; Accepted 22 October 2021; Published 22 November 2021

Academic Editor: SK Hafizul Islam

With the proliferation of intelligent services and applications authorized by artificial intelligence, the Internet of Things has penetrated into many aspects of our daily lives, and the medical field is no exception. The medical Internet of Things (MIoT) can be applied to wearable devices, remote diagnosis, mobile medical treatment, and remote monitoring. There is a large amount of medical information in the databases of various medical institutions. Nevertheless, due to the particularity of medical data, it is extremely related to personal privacy, and the data cannot be shared, resulting in data islands. Federated learning (FL), as a distributed collaborative artificial intelligence method, provides a solution. However, FL also involves multiple security and privacy issues. This paper proposes an adaptive Differential Privacy Federated Learning Medical IoT (DPFL-MIoT) model. Specifically, when the user updates the model locally, we propose a differential privacy federated learning deep neural network with adaptive gradient descent (DPFLAGD-DNN) algorithm, which can adaptively add noise to the model parameters according to the characteristics and gradient of the training data. Since privacy leaks often occur in downlink, we present differential privacy federated learning (DP-FL) algorithm where adaptive noise is added to the parameters when the server distributes the parameters. Our method effectively reduces the addition of unnecessary noise, and at the same time, the model has a good effect. Experimental results on real-world data show that our proposed algorithm can effectively protect data privacy.

1. Introduction

Big data-driven artificial intelligence (AI) has been applied to many aspects of our Internet of Things (IoT), and medical services [1, 2] are one of the most potential applications of IoT. Tremendous advances in medical technology can make human health inspections more accurate, which is also very important for taking care of the health of patients and preventing the occurrence of various diseases. In the medical Internet of Things (MIoT), "things" include doctors, patients, people who care about health, medical devices, and medicines; "network" refers to the workflow of medical treatment and health management, the process of weaving "things" into an intelligent medical "net." Due to the growth of the symbiosis of machine learning (ML) and artificial intelligence, the value of the medical IoT is increasing. Nevertheless, due to the particularity of medical data, it is extremely related to personal privacy, and the data cannot be shared, resulting in data islands. Thus, the rapid growth of medical IoT applications requires a safe and reliable learning distributed system [3].

When in various medical IoT applications, distributed machine learning is the first choice for many data processing tasks. Federated learning (FL) is the latest development of distributed machine learning, which acquires and processes data locally on the client and then transmits the updated

ML parameters to the central server for aggregation [4, 5]. The goal of FL is to fit a model generated by the empirical risk minimization (ERM) objective.

Federated learning can be traced back to federated optimization to decouple and calculate collected data on a central server [4]. Federated optimization has recently been extended to deep learning platforms, which is called federated learning [6]. Today, federated learning has become a tool of choice in many engineering fields, such as data analysis [7], speech recognition [8], autonomous driving [9], and image processing [10]. In addition, deep learning has applications in many fields of healthcare [11–13].

However, training federated learning algorithms in the field of artificial intelligence requires a large number of data samples and the need to exchange data with other devices. From a technical point of view, the introduction of computing solutions in healthcare has also caused various problems. Deep learning requires a large number of data sets, and too little data will lead to underfitting or overfitting of the neural network model. However, the data used for training may contain personal private information, such as medical records, user files, and genetic information [14, 15]. In addition, the training process of the model may also lead to the disclosure of private information. Attackers can use the correlation between the characteristics of sensitive data and model output to predict personal sensitive information based on the released model and some background information, which leads to an increase in the risk of private information leakage [16].

Designing a federated learning solution that meets privacy requirements is a challenge. The latest privacy protection scheme is mainly based on three encryption methods: secure multiparty computing (SMC), differential privacy (DP), and homomorphic encryption (HE). However, SMC is not a solution suitable for most MIoT application scenarios, because MIoT needs a noninteractive protocol to perform secure aggregation. Although HE is an effective method to prevent privacy leakage during training, the interactive decryption of HE seriously increases communication overhead. Especially for deep neural networks, a large number of hidden layers have a large number of parameters. Therefore, HE is not applicable to some medical IoT edge devices with limited computing power (such as smart bracelets and sensors).

In the past few years, many federated learning-based differential privacy has been extensively studied [17]. However, the existing methods always add the same noise in the gradient descent process, without considering the influence of data with different characteristics on the model output. In addition, privacy leaks often occur when the server distributes parameters to the model.

In order to effectively prevent information and solve the problem of data islands, this paper proposes an adaptive Differential Privacy Federated Learning Medical IoT (DPFL-MIoT) model. In this model, each client adaptively adds noise and performs local training parameters before uploading parameters to the server for aggregation. Furthermore, taking the privacy leakage of the downlink into account, noise is also added to the parameters transmitted by the server to the users participating in the training.

The main contributions of this paper are summarized as follows:

(1) We propose an adaptive Differential Privacy Federated Learning Medical IoT (DPFL-MIoT) model, which can effectively protect user privacy data and achieve a satisfactory result

(2) We present a different private federated learning deep neural network with adaptive gradient descent (DPFLAGD-DNN) algorithm based on layer-wise relevance propagation (LRP) [18]. DPFLAGD-DNN adds more noise to the features that are less relevant to the model output during training and adds less noise to the features that are more relevant to the model output. And in the stochastic gradient descent stage, the optimal step size is selected from the candidate set. When the server distributes the parameters, privacy leaks often occur, we propose differential privacy federated learning (DP-FL) algorithm, and adaptive noise is added to the parameters

(3) Extensive experiments are conducted on real-world data sets to verify the accuracy of the proposed algorithm. The evaluation results show that our algorithm can protect privacy and has good results

The remainder of this article is organized as follows. The second part is related work. The third part introduces the system model. The fourth part is background knowledge of federated learning and differential privacy and LRP algorithm. The fifth part analyzes the DPFLAGN-DNN algorithm we proposed in detail. The sixth part shows the experimental and simulation results. The seventh part is the summary of the paper.

2. Related Work

Combining different privacy protection methods with deep learning is a challenging task, and many scholars have conducted research on it.

In [6], McMahan et al. proposed a distributed training method that injects noise into the gradient to update the parameters and protect the private information of the neural network model. But in this method, the size of injected noise is accumulated in proportion to the number of training times and the number of parameters. Therefore, it may consume a large amount of privacy budget, because the number of training iterations and the number of parameters shared between multiple parties are usually very large.

Google first proposed the concept of federated learning [19]. In federated learning, participants store all training data locally, train the model locally, and then upload the parameters to the server to update the parameters. Other participants can download the updated parameters to their own devices to improve the accuracy of the local model.

A natural way to prevent information leakage is to add artificial noise, which is called differential privacy (DP) technique [17]. Considering the wide applicability of differential

privacy in deep learning models, differential privacy can also be well used for privacy protection in federated learning.

Existing DP-based learning algorithms include local differential privacy (LDP) [20–22] and DP-based distributed SGD [23]. In LDP, each client perturbs its information locally and only sends a random version to the server, thereby protecting the client and server from the leakage of private information. The work in [21] proposed a solution to establish an SGD compliant with the LDP standard, which provides impetus for various important ML tasks. The work in [22] considered the distributed estimation of the data uploaded by the client by the server and used LDP to protect these data. The work in [23] improved the calculation efficiency of DP-based SGD by tracking the detailed information of privacy loss and obtains an accurate estimate of the overall privacy loss. Abadi et al. [24] used a Gaussian mechanism to perturb the gradient and provide a more rigorous method (for example, time accounting) to track the entire privacy loss. Wang et al. [25] proposed to build an LDP-compliant SGD solution, which provides power for various important machine learning tasks. Wang et al. [26] considered the distributed estimation of the uploaded data by the server while using local differential privacy LDP to protect these data.

Committed to capturing the trade-off between privacy and aggregation performance during the training process, Geyer et al. [27] proposed an FL algorithm based on protecting customer privacy. The algorithm can obtain good training performance at a given privacy level, especially when there are enough participating clients. However, the abovementioned DP-based FL design work did not consider privacy protection in the parameter upload stage, that is, when uploading the training results to the server, the client's private information may be intercepted by hidden opponents. Similar to [27], Wei et al. [28] proposed a differential privacy federated learning framework, adding noise to the server and user local training at the same time, and detailed exploration of different privacy protection levels, the number of clients participating in training, and gradient clipping.

3. System Model

This section will propose a federated learning medical IoT based on differential privacy and layer-wise relevance propagation. Our DPFL-MIoT model is shown in Figure 1.

3.1. Model Overview. It can be seen from Figure 1 that our model is mainly composed of four parts, namely, medical cloud server, medical institution, doctor, and user. The functions of these four parts are described in detail as follows.

(1) *Medical Cloud Server*. The medical cloud server is responsible for distributing model parameters and model aggregation.

(2) *Medical Institutions*. Medical institutions are hospitals. They have their own databases and IoT equipment. At the same time, they store a large number of patients' medical data. The protection of these data is extremely important for patients.

(3) *Doctors*. Doctors hold part of patient data and provide necessary disease diagnosis and treatment support.

(4) *Users*. Users may have many IoT devices, such as bracelets and mobile phones. These IoT devices are the carriers of user data in the model and also participate in the training of the model.

3.2. Model Principle. Different from the traditional differential privacy federated learning framework, each client (entities participating in training) in our DPFL-MIoT adaptively adds noise by our proposed DPFLAGD-DNN algorithm, and considering the privacy leakage of the downlink, noise is also added to the parameters sent by the server to the users participating in the training in our proposed DP-FL algorithm. The detailed running process of the model is as follows:

(1) *Initialize Model Parameters*. First, the underlying users establish communication with the medical cloud server. The medical cloud server distributes the initial parameters of the system according to the amount of data provided by medical institutions, doctors, and users for training the model.

(2) *User Local Training*. Each client performs model training according to the parameters obtained from the medical cloud server and adaptively adds noise by DPFLAGD-DNN. After the local training is over, the parameters are uploaded to the server for aggregation.

(3) *Parameter Aggregation*. The medical cloud server aggregates parameters according to the amount of data used by the users participating in the training to train the model by DP-FL. After parameter aggregation, the parameters are distributed to users.

(4) Repeat steps 2 and 3 until the model achieves an ideal effect

4. Preliminaries

In this part, we will introduce the background knowledge of federated learning, differential privacy, and LRP algorithm.

The notations used in this paper are listed in Table 1.

4.1. Federated Learning. Federated learning [19] is an advanced distributed privacy protection machine learning technology that enables edge nodes to collaboratively train a shared global model without uploading private local data to a central server. Now consider a general FL system consisting of a server and N clients. D_i represents the local data set held by client C_i, where $i \in \{1, 2, \cdots, N\}$. On the server, the goal is to learn the data model retained on the N-related clients. When a client participates in local training, he needs to find a parameter w of the FL model to minimize a loss

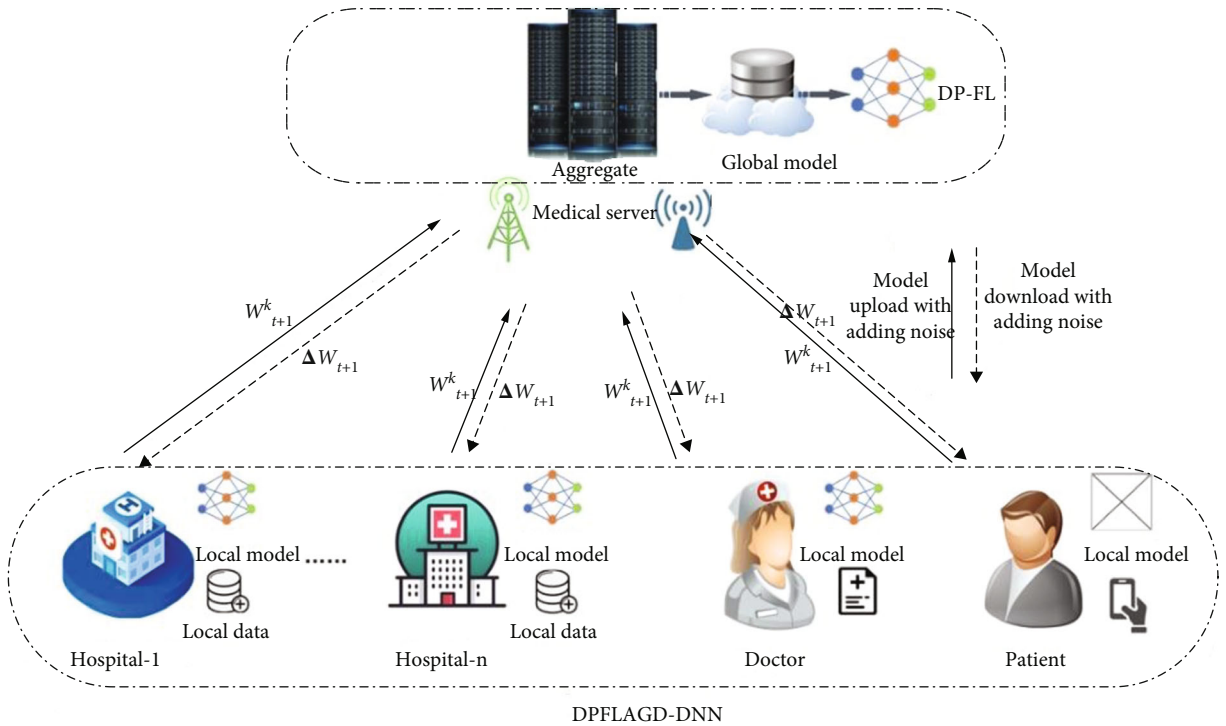

Figure 1: DPFL-MIoT model overview.

Table 1: Symbol table used in this paper.

Notation	Description
D	Data set
(x_i, y_i)	i-th record of data set D
w	Neural network parameter
w^*	Optimal parameters of NN
ε	Privacy budget
R_j	Average correlation
Δ	Sensitivity
Lap$(\cdot)$	Laplace distribution
$g(x_i)$	Gradient
β	Correlation ratio

function. Formally, the server aggregates the weights received from N clients into

$$w = \sum_{i=1}^{N} p_i w_i. \tag{1}$$

Among them, w is the parameter vector trained on the first client, is the vector aggregated on the server side, is the number of clients, and here is the size of all data samples. Such an optimization problem can be formally defined as

$$w^* = \arg \min_{w} \sum_{i=1}^{N} p_i F_i(w, D_i), \tag{2}$$

where $F_i(\cdot)$ is the loss function of the first client. Generally speaking, the local loss function $F_i(\cdot)$ is given by the local empirical risk. The training process of an FL system generally includes the following four steps:

Step 1: local training. All participating customers calculate training gradients or parameters locally and then send the local training ML parameters to the server;

Step 2: model aggregation. The server safely aggregates the parameters uploaded by the client without learning local information;

Step 3: parameter broadcast. The server broadcasts the aggregated parameters to the client;

Step 4: model update. All customers update the corresponding model with aggregated parameters and then test the performance of the updated model.

In the FL process, clients learn ML models collaboratively with the help of cloud servers. After a sufficient number of local training and update exchanges between the server and its related clients, the optimization formula (2) can converge to the solution of the global optimal learning model. Users participating in model training can use the model locally.

4.2. (ϵ, δ) – DP. A condition of (ϵ, δ) – DP [17] is that the data sets are adjacent data sets. If $|(D \setminus D') \cup (D' \setminus D)| = 1$, then the two data sets are adjacent data sets. In other words, D and D' at most differ by one record. Among them, $\epsilon > 0$ is the privacy budget, and δ represents the slack term. For any given δ, ϵ is negatively correlated with noise. We will formally define DP as follows.

Definition 1 ((ε, δ)-DP [17]). Given a random function M, if the output $S(S \in \text{Range}(K))$ of the function M satisfies the following inequality on the given adjacent data set,

$$\Pr\left[M(D_i) \in S\right] \leq e^{\varepsilon} \Pr\left[M\left(D'_i\right) \in S\right] + \delta. \tag{3}$$

Then, the function M satisfies (ε, δ)-DP. In (3), δ is the relaxation factor. Then, the random function M gives pure differential privacy. If $\delta > 0$, M gives differential privacy. The former provides stronger privacy protection than the latter. ε used to balance privacy protection and data availability. The smaller the value, the higher the level of privacy protection, and the lower the data availability, vice versa.

The implementation of differential privacy technology needs to add noise, which is closely related to the global sensitivity of the data set.

For numerical data, the Gaussian mechanism defined in [17] can be used to ensure (ε, δ)-DP. According to the literature [17], we proposed the following DP mechanism by adding artificial Gaussian noise.

In order to ensure that the given noise distribution $n \sim N(0, \sigma^2)$ is maintained (ε, δ)-DP, where N represents the Gaussian distribution, for $\varepsilon \in (0, 1)$, we choose the noise scale $\sigma \geq c\Delta s/\varepsilon$ and constant $c \geq \sqrt{2 \ln (1.25/\delta)}$. In this result, n is the value of the noise sample added to the data set, Δs is the sensitivity of the function given by $\Delta s = \max_{D_i, D_i{}'} \| s(D_i) - s(D_i{}') \|$, and s is a real-valued function.

In view of the above DP mechanism, choosing an appropriate noise level is still an important research issue, which will affect the privacy protection level of the client and the convergence speed of the federated learning process.

There are some lemmas for differential privacy. In differential privacy deep learning, we can use the following lemmas:

Lemma 2 Postprocessing [29]. *Any output calculation under differential privacy will not increase privacy loss.*

Lemma 3 Sequence combination theorem [29]. *The serialized combination of the differential privacy mechanism still satisfies the differential privacy protection.*

4.3. Layer-Wise Relevance Propagation (LRP). Layer-wise relevance propagation (LRP) [18] is an algorithm designed to calculate the correlation between each input feature and the model output.

Definition 4 (Correlation decomposition) [18]. Given a hidden layer $h_1, h_2, \cdots, h_l$, let $R_p^{(l)}(x_i)$ denote the correlation between the neurons p in the layer l and the output $f_{x_i}(\omega)$ of the model. The information sent from neuron p to its input neuron q is defined as $R_{q \longleftarrow p}^{(l-1,l)}(x_i)$. All the correlations of all the neurons in the upper layer determine the correlations of the neurons in the lower layer

$$R_q^{(l-1)}(x_i) = \sum_{p \in h_l} R_{q \longleftarrow p}^{(l-1,l)}(x_i). \tag{4}$$

The correlation decomposition is given by the following formula:

$$R_p^{(l)}(x_i) = \begin{cases} z_{pm}/z_m + \varphi f_{x_i}(\omega), z_m \geq 0 \\ z_{pm}/z_m - \varphi f_{x_i}(\omega), z_m \leq 0 \end{cases}. \tag{5}$$

Among them, parameter 7 is a predefined stabilizer used to overcome the unboundedness of and correlation. In addition, it is the affine transformation of the neuron, which is defined as

$$\begin{aligned} z_{qp} &= a_q \omega_{qp}, \\ z_p &= \sum_q z_{qp} + b_p, \end{aligned} \tag{6}$$

where a_q is the value of neuron q, ω_{qp} is the weight between neuron q and neuron p, and b_p is a bias term.

In order to realize the backward propagation of the correlation, the correlation between the neurons in the last hidden layer and the model output should be derived. Given the output variable m, the calculation method of the correlation $R_p^l(x_i)$ is as follows:

$$R_p^{(l)}(x_i) = \begin{cases} z_{pm}/z_m + \varphi f_{x_i}(\omega), z_m \geq 0 \\ z_{pm}/z_m - \varphi f_{x_i}(\omega), z_m \leq 0 \end{cases}. \tag{7}$$

By using formulas (4), (5), and (7), the correlation and input characteristics of each hidden neuron can be obtained [18]. The following equation holds:

$$f_{x_i}(\omega) = \sum_{p \in h_l} R_p^{(l)}(x_i) = \cdots = \sum_{x_{ij} \in x_i} R_{x_{ij}}(x_i). \tag{8}$$

Among them, $P_{x_{ij}}(x_i)$ is the correlation between the input feature x_{ij} and the model output $f_{x_i}(\omega)$.

5. Federated Learning with Adaptive Differential Privacy

The DPFLAGD-DNN proposed in this paper has the following two contributions:

Different from the traditional framework, in terms of Laplacian mechanism and LRP algorithm, our framework is realized by adaptive perturbation gradient according to the correlation between different features and model output.

In the global DP-FL, the server sets different privacy budgets for the users according to the data provided by the users for training the model and adds reasonable noise to

Input: Training dataset, loss function $L(\theta)$, privacy budget ϵ, gradient normal clipping S, budget increase rate α
Output: Weight parameters w_t
1: Receive w from server
2: Initialization: $t = 1$, random
3: for $j \in [1, d]$ do
4: Compute the average relevance $R_j(D)$
5: Compute the relevance ratio $\alpha_j = |R_j| / \sum_{j=1}^{d} |R_j|$
6: Calculate the privacy budget $\varepsilon_j = \beta_j \times \varepsilon$
7: $\bar{x}_{ij} \longleftarrow x_{ij} + (1/L)\mathrm{Lap}(\Delta_{h_0}/\varepsilon_1)$
8: $g_t(x) \longleftarrow \nabla_{\theta_t} L(\theta_t, x_{ij})$
9: $g_t(x) \longleftarrow g_t(x)/\max\left(1, \|g_t(x)\|_2/S\right)$
10: end for
11: while $\varepsilon \geq 0$
12: $\varepsilon \longleftarrow \varepsilon - \varepsilon_n$
13: $\varepsilon_n \longleftarrow (1 + \alpha)\varepsilon_n$
14: $i \longleftarrow \mathrm{LapNoise}(\Theta, C, \sqrt{2\varepsilon_n})$
15: $w_{t+1} \longleftarrow w_t - \eta_i g_t$
16: $t \longleftarrow t + 1$
17: end while
18: return w_t

Algorithm 1: Different private federated learning deep neural network with adaptive gradient descent (DPFLAGD-DNN).

the parameters distributed to the users in the downlink. Therefore, DP-FL can solve the problem of data imbalance and is superior to traditional methods.

5.1. User Parameter Update. In the user's local model update, we use noise disturbance to achieve differential privacy protection.

In order to design an SGD algorithm that satisfies differential privacy, a certain amount of noise is added to the gradient update process. However, adding the same noise to the gradient will affect the effect of the model. For this reason, we propose the DPFLAGD-DNN algorithm, which firstly adds noise adaptively according to the correlation between different input features and model output. Secondly, at the beginning of the loop, the loss function needs a lot of iterations before convergence, so adding noise at this time does not have a great influence on the direction of gradient descent.

However, as the optimization step proceeds, the global model gets closer and closer to the optimal value, and the direction of the gradient descent begins to become precise. At this time, the slight noise will have a great influence on the direction of noise reduction. Therefore, as the number of iterations increases, the noise should be reduced accordingly. At the same time, the noise reduction can also prevent users from running out of privacy budget, leading to premature launch of training.

Step 1 (lines 1 and 2). The parameter w is received from the server and the loop is initialized.

Step 2 (lines 3-6). Calculate the average correlation between the input feature j and the specific layer $R_j(D)$.

$$R_j(D) = \frac{1}{D} \sum R_{x_{ij}}\left(x_j\right). \tag{9}$$

In order to guarantee $R_j(D) \in [0, 1]$, each $R_j(D)$ is normalized to $R_j(D) - \gamma/(\kappa - \gamma)$, where κ and γ, respectively, represent the maximum and minimum values of the set $\{R_1(D), R_2(D), \cdots, R_d(D)\}$. We propose a correlation ratio to adaptively add noise, so that features with less correlation with the model output are added more noise and vice versa. For the features j in a specific layer, we have

$$\beta_j = \frac{\left|R_j\right|}{\sum_{j=1}^{d} \left|R_j\right|}. \tag{10}$$

The privacy budget is given by the formula below.

$$\varepsilon_j = \beta_j \times \varepsilon. \tag{11}$$

Step 3 (lines 7-10). Finally, add Laplace noise $\mathrm{Lap}(\Delta/\epsilon_j)$ to the input feature. Algorithm 1 is the pseudo code of the DPFL framework we proposed. Then, execute the gradient descent algorithm on the client

$$g_t(x, b) \longleftarrow \nabla L(w; b), \tag{12}$$

where g represents the gradient of x in the t round of update and b is the batch size. Then, we clip the gradient

$$g_t(x; b) = \frac{g_t(x; b)}{\max\left(1, \|g_t(x; b)\|_2/S\right)}. \tag{13}$$

Gradient clipping ensures that the second normal form of the gradient is limited to the range S. S is the global sensitivity.

Input: a set of step size candidates Θ, global sensitivity Δf, privacy budget ε.
Output: The index i of the best step size.
1: $\tilde{\Theta} = \{\tilde{v}_i = v + \text{Lap}(\Delta f/\varepsilon)\colon v \in \Theta\}$
2: return arg max $i \in [|\Theta|]$

Algorithm 2: LapNoise($\Theta, \Delta, \varepsilon$).

Finally, add Laplace noise Lap to the input feature.

$$\bar{x}_{ij} \longleftarrow x_{ij} + \frac{1}{L}\text{Lap}\left(\frac{\Delta_{h_0}}{\varepsilon_1}\right). \tag{14}$$

Step 4 (lines 11-18). We have customized a set Θ of step size. Each element is a loss function value, and the step size can be preset. Then, we use the LapNoise function (Algorithm 2) to select the best steps to resize. Perform gradient descent when the optimal step size is obtained.

Algorithm 2 introduces the LapNoise function. Taking candidate Θ, global sensitivity Δf, and privacy estimate ε as inputs, the algorithm returns the index i of the best step size. The function $\text{Lap}(\Delta f/\varepsilon)$ represents a Laplace distribution with an average value of 0 and a scale parameter of $\Delta f/\varepsilon$, where η is the learning rate, that is, the step size. Finally, the user's updated modulus parameter w_t can be sent to the server.

Theorem 5. *Algorithm 1 satisfies ε – differential privacy.*

Proof. Before the hidden layer node h in the neural network applies the activation function affine transformation output, the input of the node can be regarded as the linear input of the previous layer.

$$h_{x_i}(W) = b + x_i W^T, \tag{15}$$

where b is the bias term and W is the parameter of a specific hidden layer h. Given the training batch L, h can be expressed as

$$h_L(W) = \sum_{x_i \in L}\left(b + x_i W^T\right). \tag{16}$$

Given each hidden layer node $h_L(W_h)$, we add Laplace noise to the bias term b and the input feature x_i.

In lines 12 and 13, by adding adaptive noise, each input feature x_{ij} of each hidden neuron in the hidden layer h_L is perturbed. In lines 12 and 13, each input feature x_{ij} of each hidden neuron h_0 in the layer is perturbed by adding an adaptive Laplacian noise $1/|L|\text{Lap}(\Delta_{h_0}/\varepsilon_j)$.

The bias term b in the neural network can be regarded as the 0th input feature in the parameter matrix, for example, it can be expressed as x_{i0}, and for each h_L, it can be expressed as

$$h_L(W) = \sum_{j=0}^{d}\left[\sum_{x_i \in L} x_{ij} + \text{Lap}\left(\frac{\Delta_{h_0}}{\varepsilon_j}\right)\right]W^T = \sum_{j=0}^{d}\Phi L_j W^T. \tag{17}$$

Among them, we make

$$\Phi_j = \left[\sum_{x_i \in L} x_{ij} + \text{Lap}\left(\frac{\Delta_{h_0}}{\varepsilon_j}\right)\right]. \tag{18}$$

Setting Δ_{h_0} is based on the maximum value of all input features x_{ij}, and β_j can be considered as the proportion of input feature j's contribution to hidden layer neuron $h \in h_L$ to Δ_{h_0}, where d is the number of features of each tuple $x_i \in D$. All neurons h_L in layer h have been disturbed, and the following formula can be obtained.

$$\Pr\left(h_L(W)\right) = \prod_{h \in h_L}\prod_{j=0}^{d}\exp\left(\frac{\varepsilon_j\left\|\sum_{x_i \in L} x_{ij} - \Phi_j\right\|}{\Delta_h}\right). \tag{19}$$

$h_L(W)$ is the output.

$$\begin{aligned}
\frac{\Pr\left(h_L(W)\right)}{\Pr\left(h_L{}'(W)\right)} &= \frac{\prod_{h \in h_L}\prod_{j=0}^{d}\exp\left(\varepsilon_j\left\|\sum_{x_i \in L} x_{ij} - \Phi_j\right\|_1/\Delta_{h_0}\right)}{\prod_{h \in h_L}\prod_{j=0}^{d}\exp\left(\varepsilon_j\left\|\sum_{x_i' \in L'} x_{ij}{}' - \Phi_j\right\|_1/\Delta_{h_0}\right)} \\
&\le \prod_{h \in h_L}\prod_{j=0}^{d}\exp\left(\frac{\varepsilon_j}{\Delta h_L}\left\|\sum_{x_i \in L} x_{ij} - \sum_{x_i' \in L} x_{ij}'\right\|_1\right) \\
&\le \prod_{h \in h_L}\prod_{j=0}^{d}\exp\left(\frac{\varepsilon_j}{\Delta h_L}\max_{x_n \in L}\left\|x_{nj}\right\|_1\right) \\
&\le \prod_{h \in h_L}\prod_{j=0}^{d}\exp\left(\frac{\varepsilon_j}{\Delta_{h_0}}\right) \\
&\le \prod_{h \in h_L}\prod_{j=0}^{d}\exp\left(\varepsilon\frac{d \times |R_j|/\sum_{j=1}^{d}|R_j|}{\Delta_{h_L}}\right) \\
&\le \exp\left(\varepsilon_2\frac{\sum_{h \in k_L} d\left[\sum_{j=1}^{d}|R_j|/\sum_{j=1}^{d}|R_j|\right]}{\Delta_{h_0}}\right) \\
&= \exp\left(\varepsilon\right).
\end{aligned} \tag{20}$$

Therefore, Algorithm 1 satisfies ε – DP. □

Input:
Number of users K, rounds of communication T, privacy parameters set $\{\varepsilon\}_{k=0}^{K}$, number of users participating in each epoch of communication n.
Output:
The weight parameters w_T of the server model.
//Server executes.
1: Random initialize w_0
2: for $t = 0, 1, 2, \cdots, T$ do
3: $C_t \longleftarrow$ random select n users.
4: for $k \in C_t$ in parallel do
5: $w_{t+1}^k \longleftarrow$ UserUpdate$(\varepsilon/T, w_t)$
6: Send the updated parameters w_{t+1}^k to Server
7: end for
8: $w_{t+1} \longleftarrow w_t + 1/n(\sum_{k=1}^{K} \Delta w_{k+1}^k)$
9: $\tilde{w}_{t+1} \longleftarrow w_{t+1} + n_D$
10: The server broadcasts global parameters to all participants
11: end for
12: return $\tilde{w}_{t+1}$
//Users execute
13: function UserUpdate(ε, w_t)
14: $\widehat{w} \longleftarrow w_t$
15: $w =$DPAGD-DNN(ε, w_t)
16: $\Delta w_{t+1} = w - \widehat{w}$
17: return Δw_{t+1}

Algorithm 3: Differential private federated learning (DP-FL).

Theorem 6. *Algorithm 2 satisfies ε – differential privacy.*

Proof. Privacy loss will accumulate in each iteration. We use the advanced differential privacy composition theorem [29] to track the privacy loss of each step of the gradient update. The composition theorem can provide a strict boundary for privacy loss. Therefore, according to Lemma 3, we only need to ensure that the privacy budget will not be exhausted. In each iteration, two operations will cause privacy loss: the "noisy" input feature (line 7), which we have shown above is consistent with differential privacy. At each gradient update, we will check whether this update will result in a privacy budget less than 0 (line 11). If a gradient update results in a privacy budget < 0, we will not perform this update. The third line controls the entire algorithm to meet different privacy. Therefore, we prove that Algorithm 2 satisfies ε – differential privacy. □

5.2. Server Parameter Aggregation. Algorithm 3 describes the entire framework of DP-FL in detail. Assuming that there are a total of K users participating in the training, in each iteration (communication between the server and the user), a random subset Z of size $n(n \le K)$ is sampled. Only users belonging to Z send model parameters to the server.

$$w_{t+1}^k \longleftarrow \text{UserUpdate}\left(\frac{\varepsilon}{T, w_t}\right). \tag{21}$$

UserUpdate function is the user's parameter update. T is the number of communications. The server averages the parameters uploaded by users belonging to Z. Then, the server only sends the model parameter w_t to users belonging to Z.

$$w_{t+1} \longleftarrow w_t + \frac{1}{n\left(\sum_{k=1}^{N} \Delta w_{t+1}^k\right)}. \tag{22}$$

In order to ensure that the downlink channel of the T round communication meets the (ε, δ) – DP, the standard deviation of the Gaussian noise n_D added by the server to the aggregation parameter w_t can be as follows:

$$\sigma_D = \begin{cases} \dfrac{2C\sqrt{T^2 - L^2 N}}{mN\varepsilon} & T > L\sqrt{N}, \\ 0 & T \le L\sqrt{N}. \end{cases} \tag{23}$$

In order to meet the requirements of the downlink channel (ε, σ) – DP, the server needs to add additional noise. Under a certain L, the standard deviation of the additional noise depends on the relationship between the number of aggregations T and the number of clients K. Intuitively, the larger T is, the greater the possibility of information leakage. The larger number of clients can help hide its private information.

The users in each iteration are randomly selected from the total number of users K. The randomization process can reduce the model training time and increase the robustness of the model. Our solution to unbalanced data is to set a different ε for each user. The noise variance σ can be calculated from ε and δ. We set the same δ for each user. The

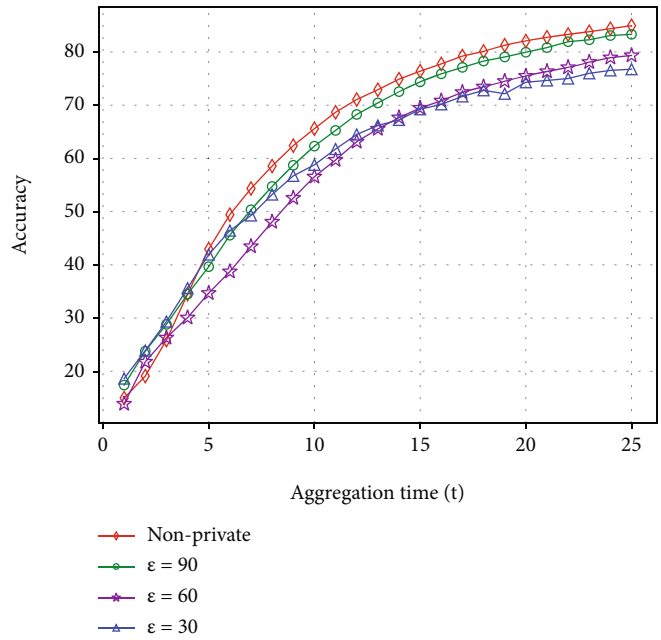

Figure 2: Comparison of the effects of 50 user models using $\varepsilon = 30, 60, 90$.

more data users use to participate in model training, the less noise is added to the input features.

Each user executes the model update algorithm locally and uploads the parameters to the server. The server averages the received parameters and sends the parameters to the user set Z. For each user, we use the DPFLAGD-DNN method to obtain local parameters. When a round of privacy budgets is exhausted, the risk of user data leakage has risen to a critical point. The user quits the joint learning training, and other users continue to train locally. We can guarantee that our DP-FL framework meets different privacy protections.

It is noting that there are many reasons why the accuracy of the model is not up to the standard, such as the amount of data involved in the training is too small, or the model is over fitted. Our method is to let users adjust parameters and increase the amount of data to continue training.

6. Experiment

6.1. Experimental Environment and Data Set. In this section, we evaluate the proposed DPFL by using neural networks and real-world federated data sets. All experiments are performed on a machine equipped with AMD Ryzen 7 5800H, 8 cores, and 16 threads, running on ubuntu 18.04. The neural network code is based on the PyTorch framework. In order to evaluate the accuracy of DPFL prediction, we use the model accuracy commonly used in deep learning as the evaluation index and change the privacy protection level, the total number of clients, and the maximum aggregation number through the control variable method. The reason why these variables are selected is that they have been widely used in other federal learning papers.

We use the Fashion-MNIST data set, which is widely used in the field of deep learning research, to evaluate the effect of the model. Fashion-MNIST is an image data set that replaces the MNIST handwritten digit set. It covers the positive pictures of a total of 70,000 different products from 10 categories. The size, format, and training set/test set division of Fashion-MNIST are exactly the same as the original MNIST, 60000/10000 training and test data division, 28×28 gray image.

Our baseline model uses a neural network with a single hidden layer containing 256 hidden units. In this feedforward neural network, we use the ReLU function as the activation function and use the SoftMax of 10 classes (corresponding to 10 classes). We set the learning rate to 0.002. We use multiple samples to evaluate this classification. Compared with many experiments exploring the effects of the DP-FL model, this setting meets the ideal conditions.

In order to reduce the experimental error without loss of generality, we will conduct each of the next experiments 10 times and take the average of the experimental results. And with the development of 5G technology, communication delay has become insignificant.

6.2. Experiment and Result Analysis

6.2.1. Protection Level Performance Evaluation. In Figure 2, we evaluate the accuracy of the model by selecting different privacy protection levels in DP-FL $\varepsilon = 30$, $\varepsilon = 60$, and $\varepsilon = 90$. In addition, we also added a baseline model to compare with our experimental results. In this experiment, we set $K = 50$, $T = 25$, and $\delta = 0.01$ and calculate the change of the model's prediction accuracy as the number of communications between the user and the server increases. As shown in Figure 2, the classification accuracy decreases

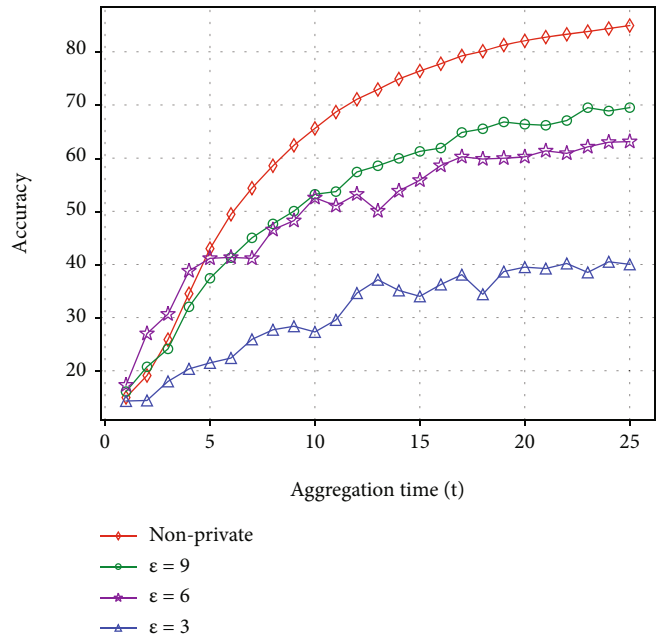

Figure 3: Comparison of the effects of 50 user models using $\varepsilon = 3$, $\varepsilon = 6$, and $\varepsilon = 9$, respectively.

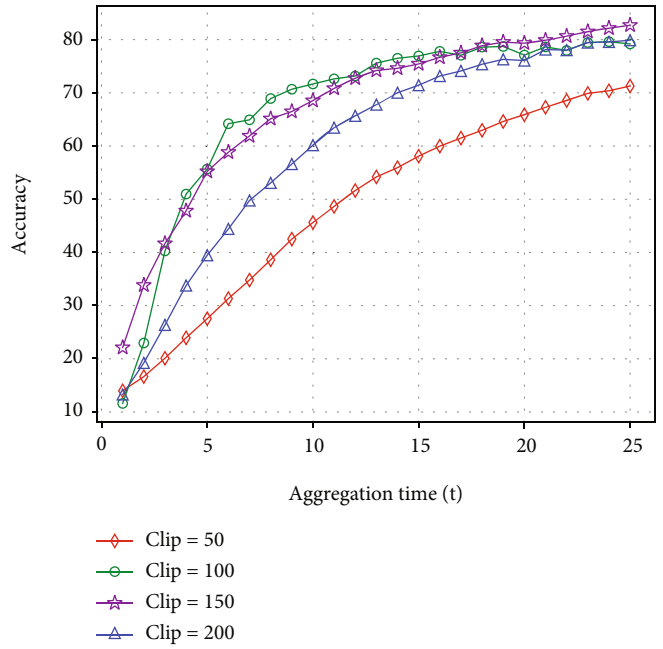

Figure 4: Different gradient clipping values of model effect comparison.

(decreases ε) as we increase the noise. Only 25 iterations are shown here. After 35 iterations, the accuracy of the model tends to converge, that is, at this time, increasing the number of iterations has a very small contribution to the accuracy.

We also conducted experiments on high-level privacy protection levels $\varepsilon = 3$, $\varepsilon = 6$, and $\varepsilon = 9$. Same as above, we set $N = 50$, $T = 25$, and $\delta = 0.01$. From Figure 3, when we increase the level of privacy protection, the value of the classification accuracy in DP-FL decreases. Moreover, it can be seen that if the added privacy noise is too large, for example, $\varepsilon = 3$, the model classification accuracy curve will fluctuate with the number of global communications, obviously because too large privacy noise seriously affects the accuracy of the model's classification results.

6.2.2. The Effect of Gradient Clipping on the Model Effect. We choose different gradient clipping values clip = 50, 000, 150 , 200 and set $\varepsilon = 30$; the number of clients $N = 50$, and the result of classification accuracy is shown in Figure 4. When

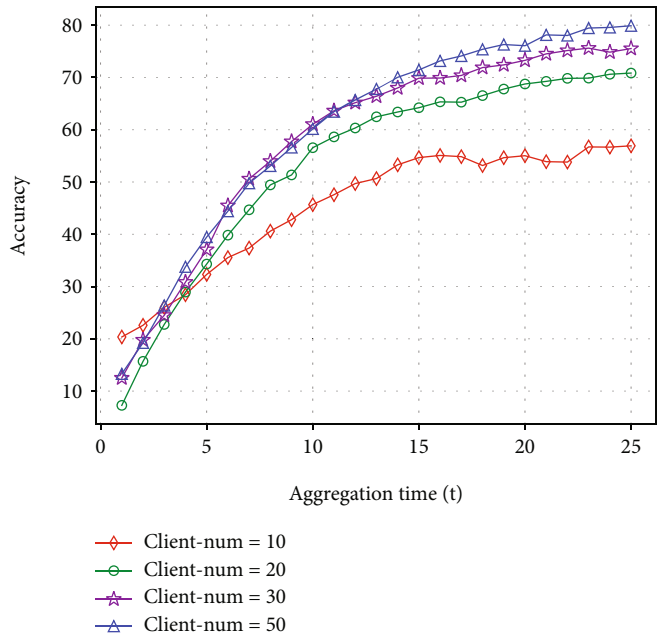

Figure 5: Model effect comparison when the client_num = 10, 20, 30, 50.

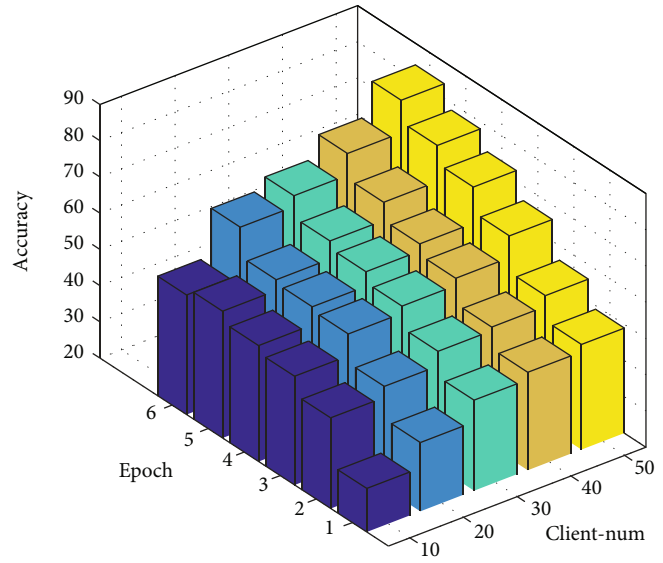

Figure 6: The relationship between the number of customers, the number of aggregations, and the accuracy.

clip = 150, DP-FL reaches convergence. We can notice that limiting the parameter norm has two opposite effects. On the one hand, if the clipping threshold clip is too small, the clipping will destroy the expected gradient direction of the parameters. On the other hand, increasing the norm limit clip forces us to add more noise parameters because it will affect sensitivity. The results of this experiment are in line with the conclusions about gradients in deep learning. In practical applications, gradient clipping is necessary, but the value of gradient clipping should be determined according to the actual situation and should not be too large or too small.

6.2.3. The Impact of the Number of Customers Participating in Model Training on Accuracy. Figure 5 shows the impact of the number of clients participating in the model training on the model performance under the conditions of $\varepsilon = 30$, $\delta = 0.01$, and clip = 100. Similar to the real-world situation, as the number of users participating in model training increases, the model performs better and better on the test set. This is because more and more customers not only provide larger global data sets for training but also reduce the standard deviation of additive noise, so that when the user sends parameters to the server, the noise added is smaller.

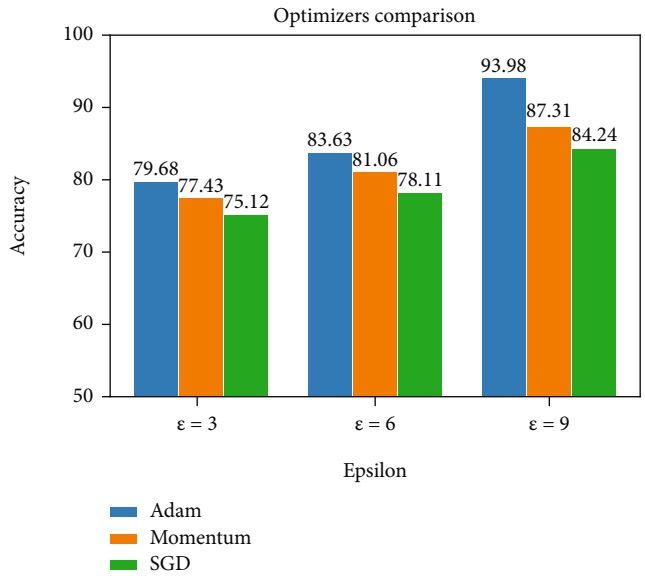

Figure 7: Comparison of optimizer effect when $\varepsilon = 3, 6, 9$, clip = 100, and client_num = 30.

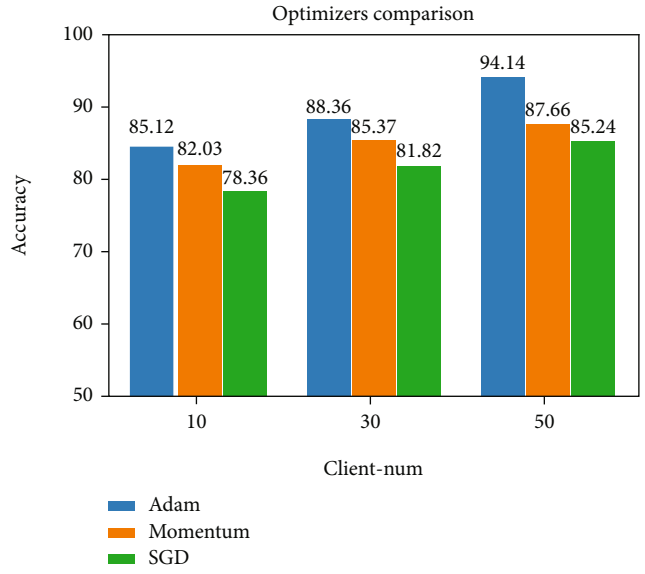

Figure 8: Comparison of optimizer effect when the client_num = 10, 30, 50, $\varepsilon = 6$, and clip = 100.

6.2.4. The Impact of the Number of Aggregations and the Number of Customers on the Effect of the Model. In Figure 6, we use the global number of communications Epoch (the number of aggregations) and the total number of clients participating in the training as independent variables and the classification accuracy as the dependent variable to explore the relationship between them. It can be seen from the three-dimensional histogram that the accuracy of the model is proportional to the number of global communications and the number of customers. One of the independent variables is fixed, and the other functional relationship is the curve described above.

6.2.5. Effects of Different Neural Network Optimizers of Customers on the Effect of the Model. The use of optimizers in neural networks is a commonly used method in the field of deep learning research. The purpose of neural network learning is to find suitable parameters to make the value of the loss function as small as possible. We have added two commonly used optimizers Adam and Momentum to the algorithm for comparison with the traditional stochastic gradient descent (SGD). Figure 7 shows the comparison results of our optimizer experiment. It can be found that when the privacy budget is 3, 6, and 9, the number of clients client_num = 30, the gradient clipping clip = 30, and the number of model iterations are 30 times, Adam and Momentum are both better than the SGD optimizer. For the name of the experimental result table, in the case of $\varepsilon = 9$, the model classification accuracy can reach 93.98%, so the Adam optimizer can be selected as the optimizer for the actual deployment of the differential privacy federated learning model.

We continue to use the controlled variable method to explore the relationship between the number of users participating in federated learning and the effect of the model. We set a common privacy budget $\varepsilon = 6$ and gradient clipping clip = 100; the number of iterations is 25, and we change the number of different clients client_num = 10, 30, 50, and the experimental results are shown in Figure 8. It can be seen that the model has a better effect.

7. Conclusions

This paper proposes a differential privacy federated learning model with adaptive noise based on correlation analysis to protect the data privacy of multiple users in the medical Internet of Things. The algorithm we proposed provides two layers of protection for participating users, namely, adding noise locally to the user and adding noise to the server side. Our method can adaptively add noise to input features according to the correlation between different features and model output. Specifically, we add more noise to the neuron gradients that are less relevant to the model's output and inject less noise into the output-related features. In addition, we optimized the neural network stochastic gradient descent algorithm, which selects the optimal step size for gradient descent according to the privacy budget. We conducted detailed experiments on Fashion-MNIST. Experimental results show that our proposed algorithm can achieve high accuracy and has certain practical application value in the field of medical Internet of Things.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Humanity and Social Science Fund of the Ministry of Education under Grant 20YJAZH078 and 20YJAZH127, Open Project of Tongji University Embedded System and Service Computing of Ministry of Education of China under Grant ESSCKF 2019-06 and ESSCKF 2019-08, and Science Education and Industry Integration Innovation Pilot Project of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2020KJC-ZD03.

References

[1] D. Połap, G. Srivastava, A. Jolfaei, and R. M. Parizi, "Blockchain technology and neural networks for the Internet of Medical Things," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 508–513, Toronto, ON, Canada, 2020.

[2] J. Cui, H. Zhu, H. Deng, Z. Chen, and D. Liu, "FeARH: federated machine learning with anonymous random hybridization on electronic medical records," *Journal of Biomedical Informatics*, vol. 117, 2021.

[3] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: the confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[4] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: wireless communication meets machine learning," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 19–25, 2020.

[5] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: a stackelberg game perspective," *IEEE Networking Letters*, vol. 2, no. 1, pp. 23–27, 2020.

[6] H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "ACM Press the 2016 ACM SIGSAC Conference," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16 - Deep Learning with Differential Privacy*, Vienna, Austria, 2016.

[7] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

[8] C. Fan, J. Yi, J. Tao, Z. Tian, B. Liu, and Z. Wen, "Gated recurrent fusion with joint training framework for robust end-to-end speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 198–209, 2021.

[9] C. De Persis and P. Tesi, "Formulas for data-driven control: stabilization, optimality, and robustness," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 909–924, 2020.

[10] W. Li, W. Wei, and L. Zhang, "GSDet: object detection in aerial images based on scale reasoning," *IEEE Transactions on Image Processing*, vol. 30, pp. 4599–4609, 2021.

[11] X. Chen, X. Lin, Q. Shen, and X. Qian, "Combined spiral transformation and model-driven multi-modal deep learning scheme for automatic prediction of tp53 mutation in pancreatic cancer," *IEEE Transactions on Medical Imaging*, vol. 40, no. 2, pp. 735–747, 2021.

[12] J. S. Heiselman and M. I. Miga, "Strain energy decay predicts elastic registration accuracy from intraoperative data constraints," *IEEE Transactions on Medical Imaging*, vol. 40, no. 4, pp. 1290–1302, 2021.

[13] J. Zucker, K. Paneri, S. Mohammad-Taheri et al., "Leveraging structured biological knowledge for counterfactual inference: a case study of viral pathogenesis," *IEEE Transactions on Big Data*, vol. 7, no. 1, pp. 25–37, 2021.

[14] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. Vincent Poor, "Scheduling policies for federated learning in wireless networks," 2019, http://arxiv.org/abs/1908.06287.

[15] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, Denver, Colorado, USA, 2015.

[16] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 739–753, San Francisco, CA, USA, 2019.

[17] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trendsin Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[18] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Muller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS One*, vol. 10, no. 7, 2015.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, in PMLR 54*, pp. 1273–1282, Fort Lauderdale, FL, 2017.

[20] A. D. Sarwate and K. Chaudhuri, "Signal processing and machine learning with differential privacy: algorithms and challenges for continuous data," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 86–94, 2013.

[21] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1054–1067, Scottsdale, Arizona, USA, 2014.

[22] X. Gu, M. Li, L. Xiong, and Y. Cao, "Providing Input-Discriminative Protection for Local Differential Privacy," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, Dallas, TX, 2020.

[23] D. Yu, Z. Zou, S. Chen et al., "Decentralized Parallel SGD With Privacy Preservation in Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5211–5220, 2021.

[24] M. Abadi, A. Chu, I. Goodfellow et al., "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, Vienna, Austria, 2016.

[25] N. Wang, X. Xiao, Y. Yang et al., "Collecting and analyzing multidimensional data with local differential privacy," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 638–649, Macao, China, 2019.

[26] S. Wang, L. Huang, Y. Nie et al., "Local differential private data aggregation for discrete distribution estimation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, pp. 2046–2059, 2019.

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

[28] K. Wei, J. Li, M. Ding et al., "Federated learning with differential privacy: algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[29] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Theory of Cryptography*, vol. 3876, pp. 265–284, 2006.