Proceedings of Machine Learning Research vol 245:1–15, 2024 2024 International Conference on Machine Learning and Intelligent Computing

Federated Learning Algorithm based on Gaussi-an Local Differential Noise

Fan Wu* WUFANGARY@163.COM
Shanghai Maritime University, College of Information Engineering, Shanghai 201306, China

Maoting Gao
Shanghai Maritime University, College of Information Engineering, Shanghai 201306, China

Editors: Nianyin Zeng and Ram Bilas Pachori

Abstract

In differential privacy-based federated learning, the data of different clients are of-ten independently and identically distributed. During model training, each client's data will optimize and converge towards its own optimal direction, causing a cli-ent drift phenomenon, resulting in a decrease in accuracy and making it difficult to obtain the optimal global model. To address this issue, a federated learning al-gorithm based on local differential privacy is proposed. Each client is assigned its own control variable ci to control the model update direction, and a global control variable c is set on the server side. The SCAFFOLD algorithm is used to aggre-gate all client model parameters and control variables. During model training, a correction term c-ci is added when updating parameters on the client side, and the model training bias is adjusted according to the global control variable obtained from the server side in the previous round, thereby controlling the model's itera-tive direction towards the global optimum. Experimental results on the CIFAR-10 datasets demonstrated the effectiveness of the new algorithm.

Keywords: federated learning; data privacy; local differential privacy; gaussian noise; model accuracy

1. Introduction

Federated learning (Rahman et al., 2021; O'Donovan, 2015; Ge et al., 2017; Shaobo Li and Roswiss, 2022) is a key technology used to address the issue of data silos. It enables data sharing and resolves data isolation while protecting user privacy. In federated learning, each client first obtains model parameters through machine learn-ing algorithms, then transmits them to a central server. The central server aggregates all received data through aggregation mechanisms and broadcasts them to all partic-ipating parties for training and prediction. This way, client data is kept on local devic-es to avoid direct leakage. However, in federated learning, passing model parameters to the central server after each iteration update can be vulnerable to extraction at-tacks and reverse inference attacks from attackers (Zhao et al., 2020; Mohassel and Zhang, 2017). Traditional federated learn-ing lacks privacy protection for transmitted parameters. Consequently, in the face of attackers' reverse inference and extraction attacks, the model parameters trained by users are more likely to be leaked in collaborative modeling.

In federated learning, the efficiency of federated learning relies heavily on the quality of aggregation algorithms used by the server. Because aggregation algorithms can address client heterogeneity and weight variations to some extent, and in central-ized federated learning, aggregation algorithms can coordinate training tasks and optimization updates. However, most aggregation algorithms lack privacy protection for users. FedAvg (McMahan et al., 2023) is the most commonly

used federated learning aggregation algorithm, which aggregates model parameters from all clients through weighted av-eraging. Although it can address client heterogeneity issues, it lacks sufficient means to address user data privacy concerns. Attackers can obtain user privacy through reverse inference attacks.

To address the privacy issues in federated learning, various solutions have been proposed. Lyu et al. (2024) and others study possible attacks on federated learning algorithms and propose privacy protection strategies such as homomorphic encryption, secure multiparty computation, and differential privacy. Sun et al. (2021) introduce the concept of Local Differential Privacy (LDP) to solve the increased information leakage risk introduced by previous federated learning and privacy protection approaches that add noise to approximate raw data. Geyer et al. (2018) propose Client-Level Federated Learning (CL-FL) based on differential privacy, where Gaussian noise is added at the server to hide the data of individual participants, which is a good choice for central-ized differential privacy methods but still susceptible to attacks from untrusted users, such as inferring client information by intercepting parameters sent from clients to the server. Kang and Ji (2022) propose Local Differential Privacy Federated Learning (LDP-FL) based on local differential privacy, using Gaussian differential privacy to add perturbation to data to resist attackers' reverse inference attacks, and design a con-straint mechanism to reduce the impact of perturbation on model accuracy. The data used for training by different clients in federated learning often does not follow an independent and identically distributed (IID) distribution. IID distribution between datasets means that data in different datasets have the same probability distribution and are mutually independent. Datasets that do not have an IID distribution are called non-IID datasets. In practical applications, although the data involved in fed-erated learning is mutually independent, the probability distributions of data between different clients are inconsistent. This leads to the convergence of training results of each client to their respective optima during model training, but the model cannot converge to the global optimum, lacking universality. Meanwhile, the performance of the model decreases when dealing with new participating parties, resulting in client drift phenomenon. Since the LDP-FL model uses the FedAvg (McMahan et al., 2023) aggregation algo-rithm, although it can address client data heterogeneity issues, the training conver-gence becomes slower and the accuracy decreases due to the difference between the training data distribution and the actual data distribution.

Therefore, a Local Differential Privacy Advanced Federated Learning (LDP-ADFL) algorithm is proposed. After training the parameters locally on the client side, Gaussian noise perturbation is added, and a parameter is introduced to control the data distribution. In the server-side aggregation algorithm, the SCAFFOLD (Karimireddy et al., 2021) algo-rithm is adopted, which not only aggregates the transmitted parameters but also in-corporates control variables transmitted from the client to maintain and control vari-ations on both the client and server sides, thereby addressing the client drift phenom-enon and ensuring global convergence to the optimum.

2. Related work

2.1. Federated learning

Federated Learning, a distributed machine learning technique pioneered by Google, operates on the principle of conducting model training across multiple data sources, each possessing local data. It consists of three main components: the central server, clients, and communication networks. Since data is kept locally on clients' devices, it provides a certain level of protection for user data. Initially, all data is collected local-ly, after which participating clients download an initialized global model

from the central server. Subsequently, each client utilizes the initialized global model for local computations. Once computations are completed, each client obtains a parameter and a new model. After local model training, the updated models and parameters are uploaded to the central server. The central server aggregates the models and parame-ters received from all clients, then distributes the new global model to each client for another round of training. This process repeats until the local models achieve the desired convergence effect. In this context, L represents the loss function, D represents the dataset, w represents model parameters, L_i denotes the local loss function of the i^{th} participant, D_i represents the dataset of the i^{th} participant, N represents the number of participants. $w*$ represents the global model parameters, and arg represents the aggregation algorithm. Currently, Federated Learning faces optimization issues such as:

$$w^* = \arg_w \min L(D, w) = \arg_w \min \sum_{i=1}^{N} L_i \left(D_i, w\right) \tag{1}$$

Generally, stochastic gradient descent is commonly used to compute the local loss function. The data aggregation algorithm determines how the global model combines local models. FedAvg (McMahan et al., 2023) is the most commonly used aggregation algorithm. Essen-tially, it is a stochastic gradient descent-based algorithm used to average the parame-ter models sent by clients. Despite its excellent performance in handling communica-tion efficiency and user heterogeneity, it cannot ensure user privacy and may en-counter client drift when training with heterogeneous data. The SCAFFOLD algorithm introduces an additional parameter to address client drift. During the client data train-ing phase, it adds a correction term based on the global control variables received from the server in the previous round to adjust its training direction, thus resolving the issue of client drift. FedBoost (Hamer et al., 2020) lgorithm employs ensemble learning to reduce communication costs between clients and servers, thereby improving communication efficiency. Additionally, by using prediction factors, it can pre-train prediction factors on publicly available data to reduce the need for user data during training. Its prima-ry focus is on addressing communication costs be-tween clients and servers, rather than solving client drift issues as effectively as the SCAFFOLD algorithm. FedProx (Li et al., 2020), like the SCAFFOLD algorithm, is an improved FedAvg al-gorithm. However, unlike SCAFFOLD, its essence lies in limiting the number of gradient updates on the client side and adding regularization terms to client training to address client drift and user heterogeneity. Adaptive Federated Optimization (Reddi et al., 2021) adjusts FedAvg's gradient up-date rules to make the data adaptable. The Secure Aggregation (Bonawitz et al., 2017) algorithm is created based on Self-Modifying Code (SMC) technology for self-regulation.

2.2. Differential Privacy

In data transmission, when attackers can obtain data before and after communica-tion in a round, they can use differentials to obtain the target user's data. Differential privacy is a measure to coun-teract this situation. For any algorithm M, let O be the set of all possible outputs of M. For any pair of adjacent datasets D and D', the algorithm M satisfies: $\Pr[M(D) \in O] \leq e' \Pr[M(D') \in O] + \delta$ then the algorithm M is said to satisfy (ε, δ) -differential priva-cy. Here, ε is the privacy protection budget, Pr is probability, $\Pr[M(D) \in O]$ represents the probability that given dataset D and output $M(D)$, the output $M(D)$ belongs to O, and δ is a relaxation term, indicating the probability of tolerating δ violating strict differential privacy. To investigate the impact of a specific feature in the data on the entire dataset, we use following statement:

$$\Delta f = \max_{x,y \in N^{|\aleph|}, \|x-y\|_1=1} \|f(x) - f(y)\|_1 \tag{2}$$

Δf is used to control noise, limiting the addition of noise through sensi-tivity to ensure that the data is not excessively distorted.

2.3. Performance Loss Constraint

Introducing local differential privacy does indeed enhance the security during the training process of federated learning. However, it also imposes a certain perfor-mance loss on the model. To address this issue, the LDP-FL algorithm proposes a design scheme, where the standard deviation of Gaussian noise should satisfy:

$$\sigma_k = \frac{2C\sqrt{2qT\ln(\frac{1}{\delta_k})}}{m\varepsilon_k} \tag{3}$$

Here, C is the gradient clipping threshold, m is the size of the local dataset, k is the client identifier, σ_k represents the variance of the k^{th} client, the sampling rate q is the fraction of i clients randomly selected from the total N participating clients for training, i.e., $q = i/N$. T is the total number of communication rounds, ε and δ are parameters related to the privacy budget and relaxation term in local differential privacy. ε represents the privacy budget, with a larger value indicating lower protection and the perturbed data being closer to the original data. δ represents the probability of tolerating a violation of strict differential privacy. Clients need to adjust the parameters C, ε, and δ according to their own situations. C defaults to the L2 norm of the gradient. The range of ε is from 0 to 10, and δ takes values between 0 and 1. If a client wants their data to have high-er confidentiality, they can decrease the value of ε and δ.

3. Method Design

Although the LDP-FL algorithm addresses client data privacy issues, it lacks measures to handle client drift phenomena. In the improved federated learning algo-rithm based on local differential privacy, we propose to address this by introducing a control variable c at the server-side and a client-specific control variable c_i at each client i. During training, we add a correction term to control the convergence direction of the model. After training the model, we add noise per-turbation to the trained model parameters. We use the constraint mechanism pro-posed in Section 1.3 to ensure that the noise does not affect the accuracy of the model training.

At the server-side, we use weighted averaging to update the gradients of all re-ceived model parameters Δw and control variable gradients Δc. The updated model parameters and control variables are then sent to all participating clients.

Regarding communication between clients and the server, by default, all clients send their parameters and control variables in parallel during one round of training. The server receives all model parameters and control variables from participating clients within a specified time frame.

In the server-side module, the following operations are performed:

a) Upon the initial system startup, initialize a global model parameter w based on the number of clients and distribute it to each participating client for joint modeling.

b) Receive the model parameter gradients and control variable gradients updated by each client, perform SCAFFOLD weighted averaging aggregation to obtain the new round of global model parameters and control variables, and then broadcast them to all participating clients in federated learning.

In the client-side module, the following operations are performed:

a) Receive the global model parameters w and control variables c sent by the server, and train the local data based on them.

b) Update the model parameters w_i and control variables c_i locally using stochastic gradient descent, and perturb the updated model parameter gradients with Gaussian noise.

c) Transmit the perturbed model parameter gradients Δw and control vari-able gradients Δc to the server.

The improved federated learning algorithm model based on local differential privacy is illustrated in figure 1.

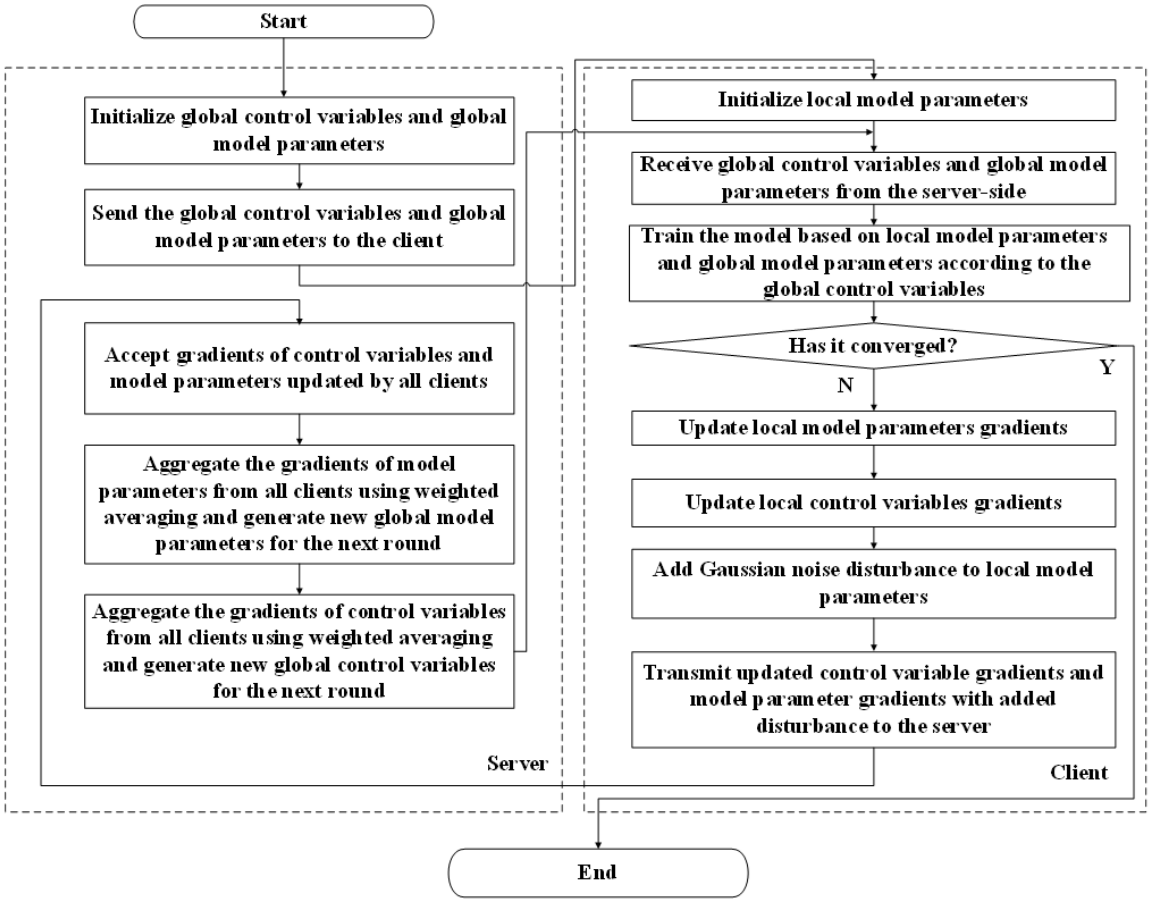

Figure 1: Improved Federated Learning Architecture Diagram.

For the algorithm of improved federated learning is shown below:

Algorithm 1 LDP_Server

Input: Initialized global model w, global control variable c, global learning rate η_{global} defined on the server, federated learning communication rounds T, number of federated learning participants k, federated learning sam-pling rate q

Output: List of model training accuracies acc_lis

Define a list of model accuracies acc_list

1: FOR t from 1 to T do

2: Sample k participants from K with sampling rate q

3: Iterate over the selected k participants:

$$\Delta w_{t,k} \leftarrow \text{LDP\_ Client}\,(k, w_t)$$

$$\Delta c_{t+1,k} \leftarrow \text{LDP\_Client}\ \ (k, c_t)$$

4: Aggregation processing:

$$w_{t+1} \leftarrow w_t + \eta_{\text{global}}(\frac{1}{K}\sum_{i \in K} \Delta w_{t,k})$$

$$c_{t+1} \leftarrow c_t + \frac{1}{K}\sum_{i \in K} \Delta c_{t,k}$$

5: Compute the model accuracy acc for this iteration

6: Append acc to acc_list

7: END FOR

8: Constrain the loss function with the performance loss constraint mechanism in Sec-tion 1.3

9: Return acc_list

Algorithm 2 LDP_Client

Input: Control variable c_t for client k, local learning rate η_{local} defined on client k, model parameters w_t obtained from the previous round of training, control variable $c_{t,k}$ obtained from the previous round, local iteration rounds N, local dataset size m, local model loss function $F_k(w)$, batch size B in sto-chastic gradient descent, gradient clipping threshold C, privacy parameters ε_k and δ_k for local differential privacy mecha-nism

Output: Model parameter gradient $\Delta w_{t,k}$, control variable gradient $\Delta c_{t,k}$ for client k

1: FOR n from 1 to N DO

2: FOR each data in training set B:

3: Compute the gradient magnitude:

$$g \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \frac{\partial F_k(w)}{\partial w}$$

4: Parameter update:

$$w_{t+1,k} \leftarrow w_{t,k} - \eta_{\text{local}} \cdot (\sum_{k=1}^{K} \frac{n_k}{n} \frac{\partial F_k(w_{t,k})}{\partial w_{t,k}} - c_{t,k} + c_t)$$

5: Compute control variable:

$$c_{t,k}^{'} \leftarrow c_{t,k} - c_t + \frac{1}{k\eta_{\text{local}}} \cdot (w_{t,k} - w_{t+1,k})$$

6: Compute noise:

$$\sigma_k \leftarrow \frac{2C\sqrt{2qT\ln(\frac{1}{\delta_k})}}{m\varepsilon_k}$$

7: Compute parameter gradient:

$$\Delta w_{t,k} \leftarrow -\eta_{\text{local}} \cdot (\sum_{k=1}^{K} \frac{n_k}{n} \frac{\partial F_k(w_{t,k})}{\partial w_{t,k}} - c_{t,k} + c_t) + N(0, {\sigma_k}^2)$$

8: Compute control variable gradient:

$$\Delta c_{t,k} \leftarrow c_{t,k}^{'} - c_{t,k} = \frac{1}{k\eta_{\text{local}}} \cdot (w_{t,k} - w_{t+1,k}) - c_t$$

9: END FOR

10: Return perturbed model parameter gradient $\Delta w_{t,k}$ and control varia-ble gradient $\Delta c_{t,k}$

Here's an explanation of the specific procedures for the client and server sides. But before that, let's briefly explain the symbols used in the text, as shown in Table 1.

Table 1: Symbol Introduction

Symbol	Description
w	Global model parameters
c	Global control variable
K	Number of clients
k	Client identifier
n	Training rounds
t	Communication rounds
η_{local}	Learning rate defined by the client
η_{global}	Learning rate defined by the server
Δw	Model parameter gradient
$\Delta w_{t,k}$	Model parameter gradient transmitted from client k in round t
Δc	Control variable gradient
$\Delta c_{t,k}$	Control variable gradient transmitted from client k in round t
n_k	Number of batches on client k
$fi(w)$	Loss function computed on each batch on client i
$Fk(w)$	Loss function on client k
$w_{t,k}$	Global model parameters received by client k in round t
$w_{i+1,k}$	Model parameters to be sent by client k in round t
c_t	Global control variable in round t
$c_{t,k}$	Control variable generated by client k in round t
$c'_{t,k}$	New control variable of client k in round t
$f(w)$	Loss function
P_k	Training data of client k
σ	Variance of Gaussian noise
ε	Privacy budget in local differential privacy
δ	Relaxation parameter in local differential privacy

Client Module. First, the data is divided into training, testing, and validation sets in a ratio of 8:1:1. With K representing the number of clients, each client $k(1 \leq k \leq K)$ initially receives the initialized global parameters w and global control variable c from the server, along with an initial control variable ck specific to itself. Since the data on each client is not identically distributed, updating model parameters w using data from each client may lead to overfitting in a distributed machine learning sce-nario. Therefore, the gradient of the original model is calculated, and the model is updated using the following function:

$$\begin{aligned} F_k(w) &= \frac{1}{n_k} \sum_{i \in P_k} f_i(w) \\ f(w) &= \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \end{aligned} \tag{4}$$

n represents the number of training rounds, n_k is the number of batches on client k, $f(w)$ is the loss function, P_k represents the training data of client k, and $f_i(w)$ is the loss function computed based on each batch of data on client $k = i$. $F_k(w)$ represents the loss function of client k. Since n_k is independent of model parameters w, the gradient of the loss func-tion can also be computed by weighted averaging, as shown in Equation.

$$\frac{\partial f(w)}{\partial w} = \sum_{k=1}^{K} \frac{n_k}{n} \frac{\partial F_k(w)}{\partial w} \tag{5}$$

For each iteration of round t, client k first receives the global parameters w_t from the server and participates in model updating. To differentiate, the global parameters involved in this update are denoted as $w_{t,k}$. The model parameters to be sent to the server are:

$$\begin{aligned} w_{t+1,k} = w_{t,k} - \eta_{\text{local}} \cdot \left(\frac{\partial f\left(w_{t,k}\right)}{\partial w_{t,k}} - c_{t,k} + c_t \right) = \\ w_{t,k} - \eta_{\text{local}} \cdot \left(\sum_{k=1}^{K} \frac{n_k}{n} \frac{\partial F_k\left(w_{t,k}\right)}{\partial w_{t,k}} - c_{t,k} + c_t \right) \end{aligned} \tag{6}$$

Here c_t represents the global control variable in round t, and $c_{t,k}$ represents the control variable generated by client k in round t. Due to the high communication cost, clients generally prefer to compute gradients accumulated over multiple rounds and then send the updated parameters to the serv-er. After local training, the control variable c needs to be updated to prevent client drift, which is done as follows:

$$c'_{t,k} = c_{t,k} - c_t + \frac{1}{k\eta_{\text{local}}} \cdot (w_{t,k} - w_{t+1,k}) \tag{7}$$

η_{local} is the learning rate defined by the client, $c'_{t,k}$ represents the new control variable of client k in round t. Since parameters may leak during transmission, leading to inference of original data by third parties, Gaussian noise with variance σ is added to the model parame-ters updated through gradient descent and transmitted to the server. Additionally, the model control parameters are updated, and the updated model parameter gradients are sent to the server as follows:

$$\begin{aligned} \Delta w_{t,k} &= w_{t+1,k} - w_{t,k} + N(0, \sigma^2) \\ &= -\eta_{\text{local}} \cdot \left(\frac{\partial f(w_{t,k})}{\partial w_{t,k}} - c_{t,k} + c_t\right) + N(0, \sigma^2) \\ &= -\eta_{\text{local}} \cdot \left(\sum_{k=1}^{K} \frac{n_k}{n} \frac{\partial F_k(w_{t,k})}{\partial w_{t,k}} - c_{t,k} + c_t\right) + N(0, \sigma^2) \end{aligned} \tag{8}$$

$$\Delta c_{t,k} = c'_{t,k} - c_{t,k} = \frac{1}{k\eta_{\text{local}}} \cdot (w_{t,k} - w_{t+1,k}) - c_t \tag{9}$$

After receiving the new parameters from the server, the model is further trained using the updated parameters until convergence.

In regard of the overfitting problem, this paper uses the control variable c as the penalty term. This variable takes part in model aggregation and changed every round from the server to avoid overfitting as well as controlling the model's iterative direction towards the global optimum.

The communication process between clients and servers proceeds as follows:

a) Initialization: Each client receives the initial model parameters w and initial control variable c from the server.

b) Local training: Clients perform model training using local data. Multiple rounds of iteration are performed locally to extract patterns and features from the data.

c) Gradient computation: After local training, clients compute the gradients Δw of the local model and gradients Δc of the control variables.

d) Adding perturbation: Clients add Gaussian noise perturbation to the computed model parameters.

e) Upload updates: Clients send the perturbed model and control variable gradients to the server.

f) Update local model: Clients receive updates to the global model w and control variables c and apply them to the local model.

g) Repeat iteration: The above steps are iterated over multiple rounds until reaching the predetermined number of iterations or meeting convergence criteria.

Through this communication process, each client conducts local model training and parameter updates, which are then aggregated with other clients to form updates to the global model. This distributed iterative process allows for collaborative learning and improvement of the model while protecting data privacy, thereby achieving the goals of federated learning. **Server Module.** In the server module, the server first initializes a global model pa-rameter w and a global control variable c, and broadcasts them to all participating clients in federated learning. Then, it waits for feedback from each client, consisting of model parameter gradients and control variable gradients, for updating. Subsequently, the server utilizes the SCAFFOLD aggregation averaging algorithm to aggre-gate the parameters received from each client:

$$\Delta w = \frac{1}{K} \sum_{i \in K} \Delta w_{t,k} \tag{10}$$

$$\Delta c = \frac{1}{K} \sum_{i \in K} \Delta c_{t,k} \tag{11}$$

K is the number of participating clients in federated learning, t represents the round, k is the client identifier, $\Delta w_{t,k}$ denotes the model parameter gradient transmitted from the kth client in round t, and $\Delta c_{t,k}$ denotes the control variable gradient transmitted from the kth client in round t. Then, the global model parameters and global control variables are updated as follows:

$$w_{t+1} = w_t + \eta_{\text{global}} \Delta w \tag{12}$$

$$c_{\text{t+1}} = c_t + \Delta c \tag{13}$$

Here, η_{global} is the learning rate defined on the server side. Final-ly, the server broadcasts to all participating clients to inform them about the new parameters for this round. In the server module, since it needs to receive model parameters from clients, it's assumed that all clients transmit data in parallel within the same training cycle. Hence, the server only needs to wait for receiving all the data, and the aggregated parameters are also sent in parallel to all participating clients.

4. Experiment and Result

To validate the effectiveness of the proposed LDP-ADFL algorithm, the experiment is divided into two parts. Experiment 1 verifies the improvement of the LDP-ADFL algorithm compared to the LDP-FL algorithm in terms of accuracy, performance loss, and addressing client drift phenomenon. Experiment 2 verifies the feasibility of the LDP-ADFL algorithm compared to LDP-FL when facing different data distributions. We use CIFAR-10 dataset for our experiment, which is a small image dataset used for recognizing general objects. It comprises 50,000 training images and 10,000 test images, with each image having dimensions of 32×32 pixels.

4.1. Evaluation Metrics

To demonstrate the effectiveness of the LDP-ADFL algorithm proposed in this paper, two main evaluation metrics are considered:

a) Global Accuracy: After multiple iterations, global accuracy is a key indicator of algorithm effectiveness. It provides an intuitive measure of algorithm performance and also reflects whether client drift is addressed. When client drift occurs, it may converge at local optima, resulting in lower global accuracy.

b) Performance Loss: Performance loss is a metric used to measure model performance, with lower values indicating better performance.

4.2. Experiment 1: Efficiency Analysis

To explore the impact of privacy budget on the performance and accuracy of the LDP-ADFL algorithm, three different privacy budget values are set: $\varepsilon = 1$, 2, and 4. Other parameters are set as $\delta = 0.001$, $q = 1$, client and server learning rates $\eta_{\text{local}} = \eta_{\text{global}} = 1$, and individual client privacy budgets $\varepsilon_i = \varepsilon$, $\sigma_i = 10^{-5}$. The experiment records the Loss and Accuracy values for each iteration round during training, and the variation curves over iteration rounds are depicted in figure 2 and figure 3.

From Figure 2, it can be observed that as the number of iteration rounds increases, the Loss value gradually decreases towards 0 and tends to stabilize. For ε =2 and 4, the LDP-ADFL algorithm outperforms the LDP-FL algorithm.

From Figure 3, it can be seen that as the number of iteration rounds increases, the Accuracy value gradually increases until reaching a certain level and then stabilizes. For ε =2 and 4, the LDP-ADFL algorithm outperforms the LDP-FL algorithm.

Comparing the results of the LDP-ADFL algorithm for ε =1, 2, and 4, it can be observed that as the privacy budget ε increases, the model's Loss value de-creases, Accuracy value increases, and performance improves. This indicates that adjusting the privacy budget can improve the model's accuracy, demonstrating the good usability of the LDP-ADFL algorithm.

By comparing the test loss of the LDP-FL and LDP-ADFL, it can be seen that LDP-ADFL is better than LDP-FL when the budget is 2 and 4, which means that when coming to the privacy budget LDP-ADFL shows its strong feasibility. That is because LDP-FL only use average method when aggregate weights, it may lead to client drift phenomenon where clients converge to local optima.

The table 2 shows the running time for LDP-FL and LDP-ADFL in 50 iterations.

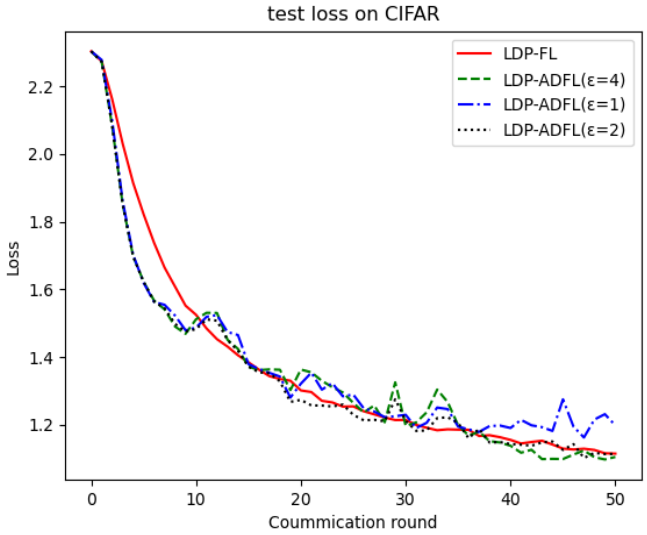

Figure 2: Test loss on CIFAR.

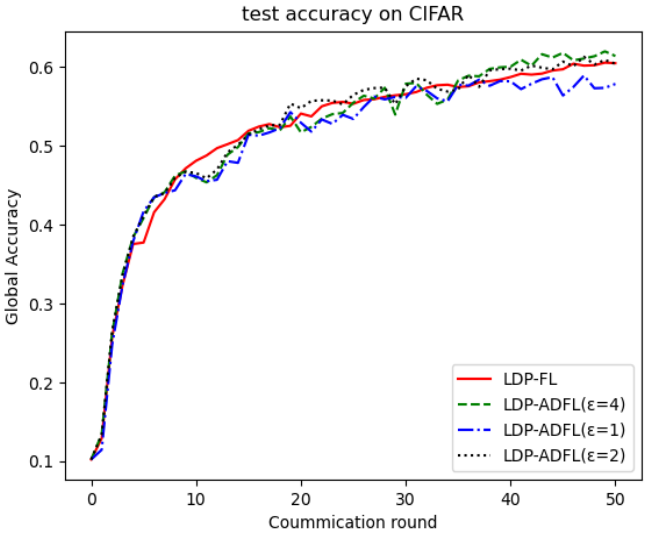

Figure 3: Test accuracy on CIFAR.

Table 2: Total running time for two algorithms

Algorithm	**Running Time**
LDP-FL	25.58
LDP-ADFL	24.69

It can be seen that LDP-ADFL is a little bit faster than the LDP-FL. This is because LDP-ADFL has add an extra term c into the model. To control the direction of the model training, control variable is needed. Since it requires a small amount of space for storage and is passed with the weight, so the communication cost is the same as the LDP-FL. The only difference is the aggregation part. Although LDP-ADFL has one more term than LDP-FL, it converged faster than average aggregation.

4.3. Experiment 2: Feasibility Analysis

To verify the heterogeneity resistance of the proposed LDP-ADFL algorithm, the average accuracy of the algorithm is tested on two types of data: Independent Iden-tically Distributed (IID) and Non-Independent Identically Distributed (Non-IID). The CIFAR-10 dataset is divided into these two categories based on labels. The LDP-ADFL algorithm is compared with the LDP-FL algorithm, and the average global accuracy values after 100 iterations on both types of data are recorded, as shown in Table 3.

Table 3: Accuracy for two different algorithms after 100 iterations

Algorithm	**IID**	**Non-IID**
LDP-FL	77.86	75.53
LDP-ADFL	78.45	80.34

From Table 2, it can be observed that the accuracy of the LDP-ADFL algorithm is higher than that of the LDP-FL algorithm on both IID and Non-IID data, with a larg-er improvement on Non-IID data. The accuracy of the LDP-FL algorithm on Non-IID data is lower than that on IID data, while the accuracy of the LDP-ADFL algo-rithm on Non-IID data is higher than that on IID data. The poorer performance of the LDP-FL algorithm on Non-IID data is attributed to the FedAvg aggregation algo-rithm's inability to handle client drift phenomena. In contrast, the SCAFFOLD aggre-gation algorithm used in the LDP-ADFL algorithm performs better in the presence of Non-IID data.

5. Conclusion

In this study, an improved federated learning method based on local differential privacy (LDP-ADFL) is proposed by adding a control variable and adopting the SCAFFOLD aggregation method to enhance the model's robustness against client drift phenomena. A noise constraint mechanism is employed to mitigate the impact of noise on model performance and accuracy. The effectiveness of the algorithm is validated on real datasets. Future work will focus on model optimization and exten-sion to the Internet of Things (IoT) environment to investigate whether global accuracy can still be improved while maintaining data privacy.

References

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

and Communications Security, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3133982.

Zhiqiang Ge, Zhihuan Song, Steven X. Ding, and Biao Huang. Data mining and analytics in the process industry: The role of machine learning. *IEEE Access*, 5:20590–20616, 2017. doi: 10.1109/ACCESS.2017.2756872.

Robin C. Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective, 2018.

Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. FedBoost: A communication-efficient algorithm for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3973–3983. PMLR, 13–18 Jul 2020.

Haiyan Kang and Yuanrui Ji. Research on federated learning approach based on local differential privacy. *Journal on Communications*, 43(10):94, 2022. doi: 10.11959/j.issn.1000-436x.2022189.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning, 2021.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Transactions on Neural Networks and Learning Systems*, 35(7):8726–8746, 2024. doi: 10.1109/TNNLS.2022.3216981.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.

Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, 2017. doi: 10.1109/SP.2017.12.

Leahy K. Bruton K. O'Sullivan D. T. J. O'Donovan, P. An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities. *Journal of Big Data*, 2:25, 2015. doi: 10.1186/s40537-015-0034-z.

K. M. Jawadur Rahman, Faisal Ahmed, Nazma Akhter, Mohammad Hasan, Ruhul Amin, Kazi Ehsan Aziz, A. K. M. Muzahidul Islam, Md. Saddam Hossain Mukta, and A. K. M. Najmul Islam. Challenges, applications and design aspects of federated learning: A survey. *IEEE Access*, 9:124682–124700, 2021. doi: 10.1109/ACCESS.2021.3111118.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization, 2021.

Chuanjiangand Li Ansi Zhang Shaobo Li, Lei Yang and Roswiss. Overview of federated learning: Technology, applications and future. *Computer Integrated Manufacturing System*, 28(07):2119–2138, 2022. ISSN 1006-5911. doi: 10.13196/j.cims.2022.07.018.

Lichao Sun, Jianwei Qian, and Xun Chen. Ldp-fl: Practical private aggregation in federated learning with local differential privacy. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1571–1578. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/217. Main Track.

Qi Zhao, Chuan Zhao, Shujie Cui, Shan Jing, and Zhenxiang Chen. Privatedl: Privacy-preserving collaborative deep learning against leakage from gradient sharing. *International Journal of Intelligent Systems*, 35(8):1262–1279, 2020. doi: https://doi.org/10.1002/int.22241.