Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Privacy-preserving federated learning based on partial low-quality data

Huiyong Wang[1,2], Qi Wang[1], Yong Ding[2,3], Shijie Tang[4*] and Yujue Wang[5]

Abstract

Traditional machine learning requires collecting data from participants for training, which may lead to malicious acquisition of privacy in participants' data. Federated learning provides a method to protect participants' data privacy by transferring the training process from a centralized server to terminal devices. However, the server may still obtain participants' privacy through inference attacks and other methods. In addition, the data provided by participants varies in quality, and the excessive involvement of low-quality data in the training process can render the model unusable, which is an important issue in current mainstream federated learning. To address the aforementioned issues, this paper proposes a Privacy Preserving Federated Learning Scheme with Partial Low-Quality Data (PPFL-LQDP). It can achieve good training results while allowing participants to utilize partial low-quality data, thereby enhancing the privacy and security of the federated learning scheme. Specifically, we use a distributed Paillier cryptographic mechanism to protect the privacy and security of participants' data during the Federated training process. Additionally, we construct composite evaluation values for the data held by participants to reduce the involvement of low-quality data, thereby minimizing the negative impact of such data on the model. Through experiments on the MNIST dataset, we demonstrate that this scheme can complete the model training of federated learning with the participation of partial low-quality data, while effectively protecting the security and privacy of participants' data. Comparisons with related schemes also show that our scheme has good overall performance.

Keywords Federated learning, Privacy protection, Low-quality data, Distributed homomorphic encryption

Introduction

In recent years, machine learning technology has greatly benefited from the development of computing processors and data acquisition methods, expanding its practical applications and improving its effectiveness [1]. Cloud computing platforms provide high-performance computing resources, enabling large-scale data processing and model training. The diversity of data also allows machine learning to be more accurate and reliable in applications such as prediction, classification, and recommendation systems. For example, the biotechnology company Berg [2] utilizes a machine learning platform to analyze extensive biological outcome data from patients (including lipids, metabolites, enzymes, and protein spectra) to emphasize critical distinctions between diseased cells and healthy cells and to identify novel cancer mechanisms.

In cases where machine learning involves multiple participants, participants are required to upload their data to a third-party server, which aims to collect large and diverse datasets for training to obtain more reliable models [3]. Most of these data are sourced from edge devices

*Correspondence:
Shijie Tang
tangsj@guet.edu.cn
[1] School of Mathematics and Computing Science, GuiLin University of Electronic Technology, Guilin 541004, China
[2] Guangxi Key Laboratory of Cryptography and Information Security, School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
[3] Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China
[4] School of Electronic Engineering and Automation, GuiLin University of Electronic Technology, Guilin 541004, China
[5] Hangzhou Innovation Institute, Beihang University, Hangzhou 310052, China

such as smartphones, glucose monitors, GPS devices, and the like. However, this data often contains participants' sensitive information, such as medical records and travel history [4]. Unauthorized access to this sensitive information could result in significant harm. Furthermore, in certain specialized industries, data sharing is not permitted [5]. Therefore, protecting the privacy of participants' data while conducting machine learning becomes of paramount importance.

Federated Learning [6] is a distributed machine learning solution proposed by Google in 2016. It aims to protect data privacy by training models on local devices. The goal is to transfer the training process of data to edge devices where participants train model parameters (gradients) on their local environments. In each iteration, participants only provide local gradient information to the server, and the cloud server aggregates the local gradients using aggregation algorithms (e.g.FedAvg) [7] to obtain updates for the global model.

While Federated Learning addresses the issue of "data silos" and mitigates privacy risks for participants, it still involves certain privacy-related concerns. Recent research has shown that malicious attackers can potentially infer sensitive participant data by analyzing the local gradients and global parameter information uploaded by participants [8, 9]. Additionally, due to varying levels of expertise and the advanced nature of data collection devices, the quality of data provided by participants may vary [10].

For example, in the context of autonomous driving [11], different vehicle models collecting road data may not guarantee global consistency, resulting in significant variations in data quality. Vehicles equipped with better perception systems often provide higher-quality data compared to those with inferior systems. Furthermore, even if a vehicle is equipped with a high-quality perception system, it does not guarantee that the collected data is absolutely complete and accurate, as error-prone perception systems or data recording processes may introduce significant errors. The involvement of a large amount of low-quality data in training can directly lead to a decrease in the accuracy of the final model or render it unusable. Therefore, introducing functionality to handle low-quality data in privacy-preserving federated learning is of practical significance.

Contributions

To the aforementioned issues, this paper proposes a privacy-preserving federated learning scheme that allows partial low-quality data to participate in training. Based on the distributed Paillier cryptosystem [12], our approach enables participants to perform offline training on their local datasets after receiving the global gradient parameters. They then engage in collaborative learning by transmitting their local gradient parameters to other participants. To prevent member inference attacks, we employ distributed Paillier homomorphic encryption to protect the transmitted gradient data. The main contributions of this work are as follows:

- We have developed an evaluation algorithm called DCEM(Data Composite Evaluation Method) to assess the quality of participant data. Based on a composite rating of data quality, we control the level of participation of data in the global iterations of Federated training, aiming to mitigate the negative impact of low-quality data on model accuracy;
- By using a single-cloud outsourcing approach, we transfer complex computations to a cloud server in order to reduce the computational burden on participants and improve the inclusiveness of the model for participants with limited computing capabilities;
- By utilizing the distributed Paillier homomorphic encryption mechanism, we have introduced a federated learning multi-party aggregation scheme that allows partial low-quality data to participate in training. This scheme effectively ensures the privacy and security of participants' private data while being user-friendly for participants to join or exit midway. Experimental results have shown that this scheme exhibits excellent overall performance.

Organization

The rest of this paper is organized as follows. Related work section gives a brief overview of related work. Preliminaries section describes the prerequisites. System model section discusses the system model and security requirements, Our proposed scheme and Security analysis sections describe the proposed scheme in detail and analyze its security. Subsequently, in Performance analysis section, the expenses and accuracy will be introduced through experiments. Finally, Conclusion section offers a summary of the entire document.

Related work

To address privacy leakage concerns in federated learning, researchers have proposed various solutions leveraging privacy protection technologies such as differential privacy [13], homomorphic encryption [14], and secure multi-party computation [15]. Haokun Fang et al. [16] proposed a federated learning scheme based on the Paillier additive homomorphic algorithm. The core idea is that all participants encrypt their local gradients that need to be uploaded using Paillier homomorphic encryption. The server aggregates these encrypted gradients

iteratively and uses the Federated multi-layer perceptron algorithm to reduce communication overhead. However, all participants use the same public key, which could potentially lead to a model being compromised by malicious participants. Jaehyoung Park et al. [17] addressed this issue by constructing a multi-key homomorphic encryption scheme to protect participants' local gradients. Since each participant holds different public and private keys, it effectively prevents malicious participation. During the iteration of global gradient updates, a joint decryption method is used, allowing the cloud server to operate in plaintext, thereby reducing the computational burden on the cloud server during the Federated training process. However, achieving collusion resistance among multiple computation servers required for joint decryption is a significant challenge.

However, none of the above-mentioned schemes have considered the issue of involving partial low-quality data in Federated training. When low-quality data excessively participates in the Federated training process described in those schemes, it can potentially result in a decrease in model accuracy or even render the model unusable. According to my knowledge, Lingchen Zhao et al. [18] was the first to consider the participation of participants holding partial low-quality data in federated learning by employing differential privacy. They achieved privacy protection by adding noise to perturb the training of neural networks and manipulating the target data. However, recent research has suggested that existing differential privacy mechanisms may not adequately balance privacy and acceptable utility in complex tasks. Yu Han et al. [19] proposed a fair incentive scheme called the Federated Learning Incentivizer, which dynamically adjusts participants' contributions based on three criteria to ensure that participants are not treated unfairly during training. However, the scheme lacks universality and fails to consider the impact of incentive strategies on data security. Guowen Xu et al. [20] proposed a method called MethIU, which uses the chi-square distribution to describe the reliability of participant data in federated learning. They employed garbled circuits and homomorphic encryption techniques to construct secure protocols for multiplication and division operations between two parties. Additionally, they improved the efficiency of the multiplication protocol. MethIU ensures that participants' reliable information is not leaked, avoiding the occurrence of training discrimination during the training process. It enables fair and equitable participation of all parties and leverages the reliability of participants to influence their involvement in global model iteration, thereby improving model accuracy. However, the dual-cloud structure of MethIU imposes significant costs on demanders, and this method is not resilient against collusion attacks between cloud servers and between cloud servers and participants. In subsequent work, Guowen Xu et al. [21] proposed a privacy-preserving single-cloud federated learning scheme (EPPFL) using the threshold Paillier homomorphic encryption mechanism. They analyzed the composition of low-quality data and employed SchUU for data evaluation, reducing the involvement of low-quality data in the Federated training process and addressing the collusion problem among servers. However, SchUU may discard some low-quality data, which can be considered unreasonable.

According to the literature review, existing federated learning schemes that support the participation of partial low-quality data while preserving privacy still face challenges regarding data integrity and privacy protection. Addressing these issues is crucial for the application of federated learning, which has motivated our research in this area.

Preliminaries

Federated learning

The data in federated learning can be classified into three types: horizontal federated learning, vertical federated learning, and Federated transfer learning [22]. This paper primarily focuses on horizontal federated learning, where participants have diverse features, but their data has little to no overlap.

Horizontal federated learning refers to the scenario where participants are distributed across different regions or organizations. Each participant possesses a private dataset, and the features among participants are diverse, and characterized by a high quantity and dimensionality. In horizontal federated learning, there may be differences in the data among participants, but their feature spaces have little to no overlap. In contrast, vertical federated learning pertains to scenarios where the feature spaces among participants overlap, but each participant possesses different sample data. On the other hand, Federated transfer learning involves participants with overlapping features and samples, but with potentially different task objectives. In this paper, we primarily focus on investigating and exploring the scenario of horizontal federated learning, where participants possess large-scale feature data but have minimal overlap in their feature spaces.

DNN (Deep Neural Network) serves various machine learning scenarios. This paper implements federated learning using a fully connected neural network. As shown in Fig. 1, DNN typically consists of an input layer, an output layer, and several hidden layers, with weights (ω) used to establish fully connected connections between neurons. Given a data set (x, y), where

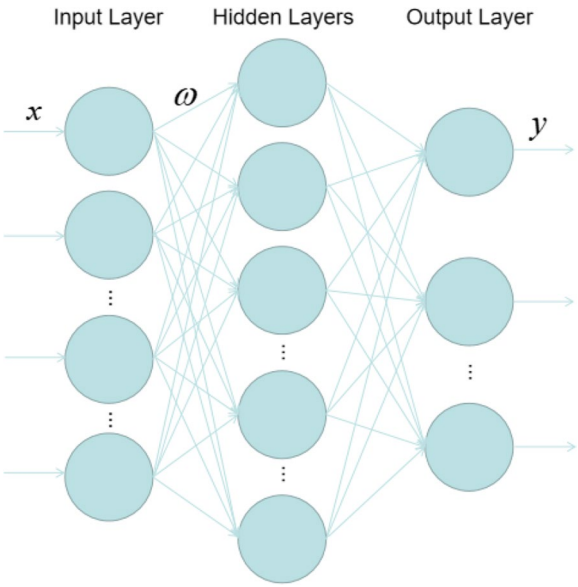


Fig. 1 Total neural network structure diagram

$x = \{x_1, x_2, x_3\}$, $y = \{y_1, y_2\}$, DNN can be used as a function to describe the relationship between an input value and a label. That is $f(x, \omega) = \hat{y}$, $\hat{y} = \{y_1, y_2\}$, DNN searches for the optimal parameter that best reflects the relationship between data and labels and make the output value wirelessly close to the real label.

In federated learning, processing a large amount of data during training for each iteration can lead to significant computational costs. To address this problem, the technique of mini-batch stochastic gradient descent (SGD) [23] can be employed, which is a common approach in previous works. Mini-batch SGD improves convergence speed and efficiency by randomly selecting a smaller subset of the data for each iteration instead of using all the data.

In mini-batch SGD, the data is strategically divided into small batches to avoid computation on the entire dataset in each iteration. This approach not only reduces computational burden but also introduces beneficial noise during the learning process, helping to avoid getting stuck in local optima and improving generalization performance. Specifically, given a training set $D = \{(x_i, y_i); i = 1, 2, ..., K\}$, we first define the loss function $L_f(D, \omega) = \left(\frac{1}{K}\right) \sum_{i=1}^{K} L_f((x, y,), \omega)$, Where $L_f\left((x, y), \omega\right) = \left\| y - f(x, \omega) \right\|_2$, and $\|\cdot\|_2$ represents the 2-norm of a vector.

In $j - th$ iteration, the participants randomly selects part D_i of the held data, where $D_i \in D$. Then participants only used the selected data D_i to participate in the training in this round of iteration and uploaded the local gradient information to the cloud server for aggregation. So in federated learning, can update the weight as:

$$\omega_{j+1} \leftarrow \omega_j - \alpha \frac{\sum_{k \in K} g_j^k}{\sum_{k \in K} \left| D_j^k \right|} \tag{1}$$

Finally, the cloud server will broadcast ω_{j+1} to each participant to iterate the participant data again until our preset convergence conditions are met.

Distributed paillier cryptosystem

Our solution utilizes the distributed Paillier cryptosystem to achieve secure aggregation, as depicted in Fig. 2. The distributed Paillier cryptosystem offers several advantages in practical applications. Firstly, by setting a threshold, we can decompose the private key and distribute partial private keys to the participants. By collecting a sufficient number of partially decrypted ciphertexts, we can aggregate them to obtain the plaintext information. Secondly, the distributed Paillier cryptosystem satisfies additive homomorphism, meaning that we can perform addition operations on the plaintext by conducting

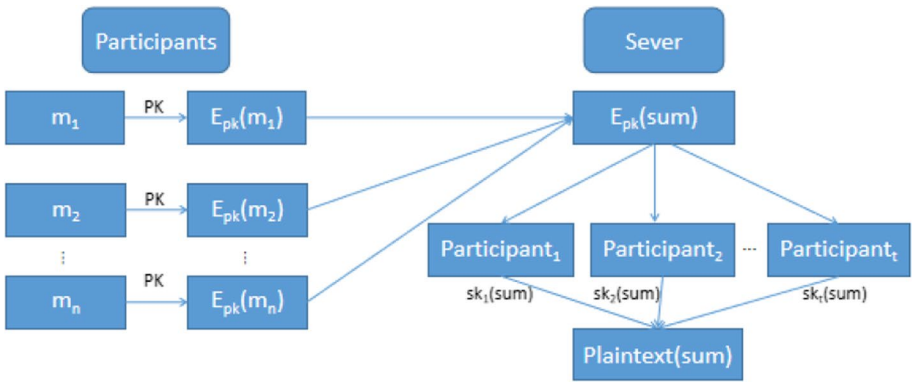


Fig. 2 "SAP" decryption diagram

multiplication operations on the ciphertext. This property allows us to protect data privacy while performing secure aggregation. By utilizing the distributed Paillier cryptosystem, we ensure secure aggregation while preserving data privacy, enabling efficient computations.

The distributed paillier cryptosystem can be divided into four parts: key generation, private key splitting, encryption algorithm and decryption algorithm.

Key generation: Let Θ be the security parameter, p and q are two randomly selected large prime numbers of the same length, calculate $N = pq$ and $\lambda = \frac{lcm(p-1,q-1)}{2}$. The public key is set to $P_k = N$ and the private key is set to $S_k = \lambda$. where *lcm* represents the calculation of the least common multiple between two numbers.

Private key splitting: Let's choose δ satisfy both $\delta \equiv 0 \bmod \lambda$ and $\delta \equiv 1 \mathrm{modN}^2$. And then we define a polynomial $q(x) = \delta + \sum_{i=1}^{k-1} a_i x^i$; where $a_1, \ldots, a_{k-1}$ is $K-1$ random numbers selected in $Z^*_{\lambda N^2}$. $\alpha_1, ..., \alpha_n \in Z^*_{\lambda N^2}$ are n different non-zero elements known by each party. Part of the private key is $s_k^{(i)} = q(\alpha_i)$, send to participant.

Encryption algorithm: For a given plaintext m we choose a random number $r \in Z_N$, ciphertext c can be generated as $c = [m]$; where $[m] = g^m \cdot r^N \mathrm{modN}^2 = (1 + \mathrm{mN}) \cdot \mathrm{r^N modN}^2$.

Decryption algorithm: Partial decryption of ciphertext c by partial private keys held by the participant;For convenience, we use $CT^{(i)}$ to represent partial decryption results. Because $gcd(\lambda, N^2) = 1$ according to the Chinese residual theorem: $\delta = \lambda \cdot (\lambda^{-1} \mathrm{modN}^2) \mathrm{mod} \lambda \mathrm{N}^2$, $CT^{(i)} = [m]^{q(\alpha_i)} \mathrm{modN}^2$.

Each user holding part of the private key partially decrypts the ciphertext, obtains the partial decryption result, and sends the result to the cloud server. After receiving the decryption result, the cloud server aggregates the plaintext by the method described in the document [24].

System model

System architecture

As shown in Fig. 3, this paper considers a system framework with two entities: participants and cloud server. Collaboration between these two entities is essential for federated learning. Specifically, before federated training started, all participants agreed to use the DNN model for training. In each iteration, participants use the received global gradients to train their local data to obtain the local gradient set, cooperate with the cloud server to calculate the composite evaluation value of the data, and then encrypt and upload to the cloud server. The cloud server obtains the local gradient data of participants and the composite evaluation value of the data, updates the global model, and broadcasts it to each participant. This process is repeated until the DNN model reaches the

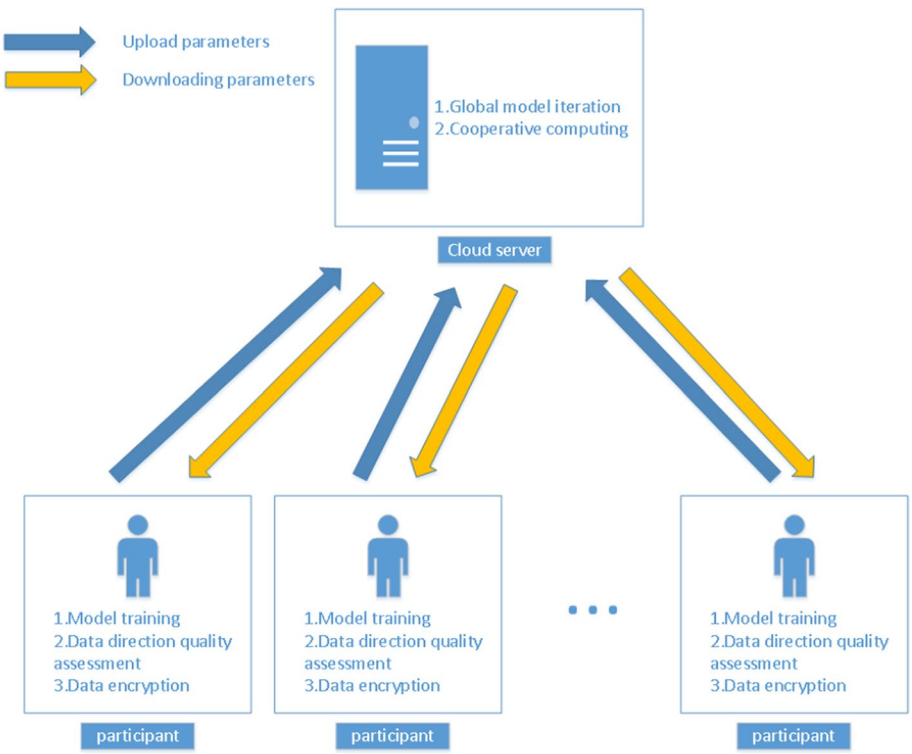


Fig. 3 System model

convergence criterion. During this iteration, participants collaborate with the cloud server by transmitting their local gradients, allowing them to jointly train a global model. This federated learning framework enables participants to share and merge their model updates while protecting data privacy and resisting the intrusion of low-quality data over participation in training, aiming to improve the overall learning performance.

During this process, the local gradients of participants are considered private data. To ensure the privacy and security of participant data, a secure aggregation framework is introduced using the distributed Paillier homomorphic encryption mechanism. Distributed homomorphic encryption allows participants to perform collaborative computations in a distributed environment, effectively protecting the privacy and security of participant data.

Remark 1

We only consider the architecture with a single cloud server, primarily achieving secure aggregation of the transmitted data from participants through cloud server outsourcing. Specifically, when the cloud server S receives the local gradient sets transmitted by each participant, it performs aggregation operations in the cloud, calculating the sum of the locally uploaded gradients in an encrypted state. This secure framework has been widely used in previous works. In recent studies on federated learning [25–27], solutions with dual-cloud or multi-cloud structures have also been proposed. However, such setups require ensuring the absence of collusion among multiple servers, posing higher demands on practical applications. In our setting, the federated learning process is accomplished solely through a single cloud server, performing all the aggregation and computations, thereby avoiding the aforementioned issues. Additionally, the use of distributed homomorphic encryption allows for tolerance towards a small number of malicious participants colluding with the cloud server, while also being inclusive of participants' mid-training joining and exiting.

Threat model and privacy objectives

In federated learning, we assume that some participants and the server act semi-honestly and treat them as potential adversaries. The threat model is defined as follows:

- Single malicious participant attack: A single malicious participant in federated training may exhibit curiosity towards the model parameters uploaded by other participants and attempt to infer their private information.
- Cloud Server Attack: In the process of global model iteration, cloud servers may launch inference attacks on participants' private data by learning and reasoning over the local parameter data uploaded by participants in federated learning. This can lead to privacy breaches and the potential for malicious actors to infer sensitive information about the participants.
- Some participants conspired with the cloud server to attack: When some malicious participants collude with the server to launch an attack, they may exploit shared model parameters to infer and obtain private information of other participants, leading to security issues such as privacy leaks among participants.

To achieve federated training without compromising the privacy of participants' data, it is necessary to achieve the following privacy protection goals:

- Privacy of participant's local gradients: Attackers may obtain the local gradient data obtained by participants during local training using their data and use this information to reconstruct the original data, leading to privacy breaches (such as medical records, location, visited websites, etc.). To protect the privacy of participant data, we set participants should encrypt the local gradient before uploading it to the cloud server, in addition to local operations. The cloud server only performs aggregation and computation on the encrypted data, ensuring the privacy of the original data.
- Privacy protection of aggregated evaluation values: The privacy of the aggregated evaluation values of each participant also needs to be ensured. If leaked, participants with lower data quality may face discrimination during the training process, thus impacting the fairness of training. Therefore, it is crucial to maintain the privacy and confidentiality of the aggregated evaluation values to uphold the fairness and integrity of the training process.

Our proposed scheme

This section provides a detailed description of the Privacy-Preserving Federated Learning with Low-Quality Data Participation (PPFL-LQDP) framework that allows participation of partially low-quality data while ensuring data privacy and security without data deletion. PPFL-LQDP enables interactions between a single cloud server and participants, supporting encryption-based operations on ciphertexts. The framework ensures the fulfillment of specified privacy requirements while preventing the adverse effects of including low-quality data excessively in the Federated training process.

Firstly, let introduce our developed DCEM framework. DCEM is capable of conducting a composite evaluation of the contributed data by participants, ensuring a reduction in low-quality data contributions during Federated training. This guarantees that the accuracy and convergence speed of model training will not be negatively affected. The framework also incorporates a key "SAP" (Secure and Accelerate Partnership) protocol that promotes secure interaction between the cloud server and participants, enabling the secure execution of DCEM operations.

Finally, we provided a detailed explanation of how the PPFL-LQDP utilizes the "SAP" protocol. This included an overview of the interaction process between the cloud server and participants, encryption of the DCEM operations performed on the data provided by participants, as well as ensuring the privacy and security of the participants' local gradient information and the composite evaluation values of data quality.

Composite evaluations containing participants with low quality data

In practical applications of federated learning, it is challenging to avoid excessive participation of low-quality data due to different data sources and varying participants' capabilities. As a result, participants' datasets may contain low-quality data, and a large amount of low-quality data participating in training, especially components that are opposite to our global optimal gradient, can lead to negative convergence of the model. Depending on the type of data, the reasons for low data quality can be defined as follows: 1) the gradient vector is in the opposite direction to our global optimal gradient, so we consider the gradient to be negatively correlated; 2) there is significant variability between local gradient data and global optimal gradient data. Both factors can lead to reduced model accuracy and slower convergence speed. To address the impact of low-quality data, it is necessary to adjust the level of data participation properly. In other words, we need to seek an evaluation method to adjust the weight of participant data in the global model.

To address the negative impact of excessive participation from low-quality data contributors in federated learning, the PPFL-LQDP framework implements a composite evaluation mechanism. This mechanism evaluates low-quality data contributors by performing DCEM operations and reduces their participation in the Federated training. Specifically, for each participant with low-quality data, the DCEM operation adjusts their gradient values using unique threshold settings and an orthogonal loss-based approach. This adjustment helps mitigate the negative impact of low-quality data while improving the overall accuracy of the model.

In this paper, we assume that all the data follows an independent and identically distributed (IID) distribution. In the Federated training process, we have K participants, and each participant performs local mini-batch gradient descent training. As a result, each participant obtains a set of gradients for their respective local data. Each participant can get a set of gradients. We assume that the local gradient of this participant is $g_k = (g_k^1, g_k^2, \cdots, g_k^Z)$. During Federated training, each participant transfers its local gradients to the cloud server for global gradient iteration. Building upon the system model, three important components of DCEM are described in detail.

Component orientation evaluation

The direction of the global gradient indeed determines the iteration speed and direction of our model, and each local gradient component has a certain influence on the global gradient. Based on the previous description, the first step is to evaluate the vector direction difference between the local gradients and the global optimal gradient. Prior research has shown that DNN models exhibit a stable convergence pattern during each iteration.

Algorithm 1 COEs

input : local gradient $g_k = (g_k^1, g_k^2, \cdots, g_k^Z)$,
global optimal gradient $g_* = (g_*^1, g_*^2, ..., g_*^Z)$.
output: y(Direction evaluation weight).

Initialize global optimal gradient;
In j-th iteration, given the global optimal gradient g_{*j-1};
For each of the participants $g_k \in g_K$ do
If $t = \cos(g_k^z, g_*^z) \geq 0$ then
$y = t^2 + 1$
Else $y = -t^2 + 1$

Specifically, we first assume that each participant has a local gradient g, and for a given global optimal gradient g_*, we evaluate the direction in which each participant holds the local gradient of the data by implementing the "COEs" algorithm. First define a piecewise function to calculate the Angle between g and g_*, as follows:

$$y = \begin{cases} t^2 + 1, & t = \cos(g, g_*) \geq 0 \\ -t^2 + 1, & \text{else} \end{cases} \tag{2}$$

Where t is the calculation result of cosine between two vectors. According to Eq. (2), we can calculate the directional difference between the local gradient and the global ideal gradient of each participant, and then output the y value to complete the component orientation evaluation.

The "COEs" component that we have developed requires all participants to execute the protocol in an

offline state, with each participant's local gradient carrying the "COEs" evaluation result. The "COEs" evaluation effectively assesses each local gradient, preventing excessive participation of low-quality data that may degrade model accuracy, without deleting any data. Additionally, since all participants' data is involved in the training, the convergence of the model is not negatively affected. In Chapter 7, we demonstrate the superiority of this approach through experiments and show that it effectively addresses the challenges posed by low-quality data in federated learning.

Component dispersion evaluation

In the previous section, participants obtained the direction evaluation of all data by executing the "COEs" protocol. Based on our definition of low-quality data, highly dispersed data is also considered low-quality. Previous research has shown that the Euclidean distance can be used to assess the data quality of participants. Specifically, the shorter the distance between the local gradient and the global optimal gradient, the higher the quality of the data can be deemed. We then assign higher ratings to influence the weight of this participant in Federated training. In this paper, we further analyze the local gradient variance by processing the distances through mean calculation. This allows for a clearer and more accurate reflection of the degree of deviation in participants' local gradients, facilitating our evaluation process. By incorporating mean analysis into the evaluation process, we can effectively measure the data quality of participants and introduce appropriate perturbations during the training process. Overall, by incorporating mean analysis into the evaluation process, we can more effectively measure the data quality of participants, enabling better assessment and appropriate perturbation during the training process.

Algorithm 2 CDEs

input : local gradient $g_k = (g_k^1, g_k^2, \cdots, g_k^Z)$,
global optimal gradient $g_* = (g_*^1, g_*^2, ..., g_*^Z)$,
global weight $w_* = (w_*^1, w_*^2, ..., w_*^Z)$,
learning rate α.
output: D_k^z (The value of the k-th participant's dispersion at the j-th iteration).

Initialize global gradient g_* and global weight w_*;
Repeat
For each of the participants $k \in K$ do
Calculate the Euclidean distance between the local gradient and the global optimal Gradient
End for
For (The distance was calculated for each participant) do
Update the local gradient dispersion evaluation value (e.g., Eqn. (3))
End for
Return D_k^z

In this paper, we establish a method called "CDEs" for low-quality screening and evaluation of training data for discreteness, as shown in protocol 2. We only consider the scenario of horizontal federated learning, where the private data held by the participants are all IID samples. Specifically, we set up a total of K participants for training, and each participant obtains a set of local gradients by executing the mini-batch gradient descent method locally. In the "CDEs" approach, each participant calculates the distance between their local gradient and the globally optimal gradient, which is then uploaded to the cloud for aggregation, resulting in an assessment of the participant's dispersion. The dispersion update for each user's data is as follows:

$$D_k^z = \log\left(\frac{\sum_{k=1}^{k=K} dis}{K \cdot dis} + 1\right) \tag{3}$$

where "dis" represents the Euclidean distance between two vectors, and the specific calculation formula is $dis(a, b) = (a - b)^2$.

When participants' data is involved in DNN training, the differences in the data can lead to a certain degree of model loss. Algorithm 1 allows us to evaluate the direction of gradient components during training which helps analyze the components with superior direction, ensuring that the local gradients contributed by participants are beneficial to the convergence of the model. Next, Eq. (3) is used to measure the dispersion and differentiate the participation weights of participant data in training the DNN. This ensures the accuracy and convergence speed of the model are reasonably maintained.

Remark 2

In previous research [28, 29], using distance to describe the quality of data held by participants has achieved good results. In our work, we further process the distance evaluation by calculating distance weights through the ratio of distance to the mean. Specifically, for the data under evaluation, we construct the ratio between the average distance between the local gradient and the optimal gradient of the training data, and the distance between the evaluated data and the optimal gradient. A larger ratio indicates a closer distance and higher data quality, resulting in a higher weight in the training process. Conversely, for lower-quality data, we set a threshold to reduce their participation. By calculating the distance weights based on the ratio, we can effectively differentiate and assign weights to data based on their quality. This enables us to include high-quality data more prominently in the training process while reducing the influence of low-quality data through the threshold

setting. This approach allows us to better manage the participation of participants with varying data quality and ensure that the overall training process is more robust and accurate.

Data evaluation aggregation

Given direction evaluation and dispersion evaluation of the training data provided by each participant, the composite evaluation Q of the data can be obtained from Eq. (4):

$$Q_k^z = D_k^z \cdot y \tag{4}$$

Remark 3

In the two previous sections, we conducted component direction quality evaluation and dispersion quality evaluation on the given data. We can perform a composite evaluation based on Eq. (4) to generate a composite evaluation value for the data. It can be observed that assigning higher weights to the training data provided by a participant, whose local gradient is closer to the expected global optimal gradient value, is reasonable. According to the state-of-the-art research [30, 31], in scenarios where participants hold independent and identically distributed (IID) samples, similar samples exhibit high consistency in direction and dispersion. However, in federated learning, participants' data differs significantly due to variations in data collection methods and processing capabilities. In this scenario, it is crucial to control the participation ratio of data in training through composite evaluation values to mitigate the negative impact of excessive involvement of low-quality data on model accuracy.

Aggregated value update

Given the combined evaluation value of Q_k^z each group of training data, the update of each local gradient component is shown as follows:

$$g_*^z = \frac{\sum_{k=1}^{k=K} Q_k^z g_k^z}{\sum_{k=1}^{k=K} Q_k^z} \tag{5}$$

Remark 4

According to Eq. (5), in the summary of participating data, data with higher comprehensive evaluation value will be given higher weight to participate in training, to ensure that the model will not be unavailable due to excessive addition of low-quality data. we filter based on

$$\left(Q_k^z - \min_{i\in[1,X]} Q_i^z\right) \Big/ \left(\max_{j\in[1,X]} Q_j^z - \min_{i\in[1,M]} Q_i^l\right) < \beta$$

where β is a preset threshold that participants collectively negotiate [11, 15, 16]. In our setup, low-quality data is not directly removed, but rather its participation is influenced through a well-designed structure to ensure the confidentiality of the data quality held by each participant. We allow the inclusion of low-quality data, but with a significantly lower weight, ensuring that the majority of the training data is of trusted quality. In the experimental section, the superiority of our approach in terms of model accuracy will be fully demonstrated.

Secure aggregation protocol(SAP)

The proposed secure aggregation scheme is achieved through the improvement of the Paillier encryption mechanism. we assume that each participant in the Federated training holds a private dataset, and we aim to protect the local gradients of all participants' data, as well as the composite evaluation values, through the "SAP" protocol. Specifically, as shown in Fig. 2, the privacy data of participants is encrypted and the ciphertext is transmitted to the cloud server S. Upon receiving the ciphertext data from all participants, the cloud server performs multiplication operations on the ciphertext using additive homomorphism and then publicly reveals the aggregated ciphertext result to all participants. Subsequently, participants perform partial decryption on the ciphertext c using their respective partial private keys and send the partially decrypted results to the cloud server S. Finally, the cloud S stops receiving decryption data until it collects the partially decrypted results from t participants then aggregates all the received partially decrypted results to obtain the sum of the local gradients of k participants.

The establishment of the PPEL-LQDP

The PPEL-LQDP established by us is shown in Algorithm 3, which mainly includes two phases, namely system setup and encrypted execution of DCEM scheme. The steps of PPFL-LQDP are explained in detail next.

Algorithm 3 Implementation of PPFL-LQDP

Setup : Given the parameter Θ, The third authority generates a set of public and private keys for all participants in the system and the cloud server based on the distributed Paillier cryptographic mechanism, where the public key is p_k and the private key group is represented by $(sk_1, sk_2, \ldots, sk_K)$. The public key p_k is broadcast to all participants and the outsourced cloud server S, part of the private key is distributed to each participant k through secret channel sk_k. Participant users and server S agree on a DNN training model to implement training. The cloud server initializes DNN global maximum gradient $g_* = (g_*^1, g_*^2, ..., g_*^Z)$ and global weight $\omega_* = (\omega_*^1, \omega_*^2, ..., \omega_*^Z)$.

Encrypted: DCEM

1 Users:
Each participant executes the mini-batch gradient descent method locally after receiving the global weight broadcast by the cloud server;

2 Each participant was trained locally to obtain the local gradient g_k and the global optimal gradient g_* to evaluate the direction of "COEs" algorithm;

3 Each participant locally calculates the euclidean distance between the local gradient g_k^z and the global optimal gradient g_*^z as $dis\,(g_k^z, g_*^z)$;

4 Each participant k encrypts the $dis\,(g_k^z, g_*^z)$ and $K \cdot dis\,(g_k^z, g_*^z)$ as $E_{p_k}\,(dis\,(g_k^z, g_*^z))$ and $E_{p_k}\,(K \cdot dis\,(g_k^z, g_*^z))$, and encrypts $-\log\,(K \cdot dis\,(g_k^z, g_\star^z))$ as $E_{p_k}\,(-\log\,(K \cdot dis(g_\kappa^z, g_\star^z)))$, and send them to the cloud server;

5 S:
For each component received, S multiplies the ciphertext $E_{p_k}\,(sum) = \prod_{k=1}^{k=K} E_{p_k}\,(dis(g_k^z, g_\star^z)) = E_{p_k} \sum_{k=1}^{k=K} dis(g_k^z, g_\star^z)$, and calculate $E_{p_k}\,(mol) = E_{p_k}\,(sum) \cdot E_{p_k}\,\left(K \cdot dis\left(g_\star^z, g_\star^z\right)\right)$. Based on our "SAP" , the cloud server gets the aggregate value as $mol = \left(mol^1, mol^2, \cdots, mol^Z\right)$;

6 Then server S encrypts $\log(mol)$ as $E_{p_k}\,(\log(mol))$. Next, the cloud server calculates a dispersion rating for each participant as $E_{p_k}(D_k^z) = E_{p_k}(\log(mol)) \cdot E_{p_k}(-\log(K \cdot dis(g_k, g_*)))$, and send the calculation result to each participant k;

7 Users:
Each participant k makes a local evaluation combination as $E_{p_k}\,(Q_k^z) = E_{p_k}\,(D_k^z)^y$, and send then to cloud server;

8 S:
S calculates $\prod_{k=1}^{k=K} E_{p_k}(Q_k^z g_k^z)$ and $\prod_{k=1}^{k=K} E_{p_k}(Q_k^z)$, and through the "SAP" to calculate the obtain $\sum_{k=1}^{k=K} Q_k^z g_k^z$ and $\sum_{k=1}^{k=K} Q_k^z$. After obtaining valid data, the cloud server calculates g_*^z (eq. (5));

9 After all g_k^z is obtained, the cloud server updates the current global optimal gradient;

10 S and users
The cloud server S and the participant repeat steps 3-9 until the convergence condition is met, and then update the global optimal gradient g*;

11 S:
Cloud server S updates the global weight ω_* (eq. (1)) iteratively;

12 S and users
Repeat steps 1-11 until the difference between the two iterations is less than the threshold set in advance.

System Setup: We first generate the public key p_k and partial private key groups $(s_{k_1}, s_{k_2}, ..., s_{k_K})$ based on the distributed paillier cryptosystem through a third-party authorization center (TA), and distribute partial private keys to each participant through a secret channel. To initiate the DNN model, the cloud server first initializes the global weights and the global optimal gradient.

Encrypted DCEM: Participants encrypt their private data using a distributed Paillier cryptosystem and then send it to the cloud server. The cloud server collaborates with the participants to perform the encrypted DCEM operations, obtaining the composite evaluation values of the data. For simplicity, we use $P_k(a)$ to represent the ciphertext of plaintext a. As shown in Algorithm 3, an encrypted DCEM consists of five main parts: 1) component orientation evaluation. 2) component dispersion evaluation. 3) data evaluation aggregation. 4) encrypted aggregated Value Update. 5) weight update. The detailed techniques for each phase are shown below.

(1) Component orientation evaluation: In this stage, steps 1 and 2 are performed to evaluate the direction of the data. First, after receiving the global weight and optimal gradient initialized by the cloud server, each participant $k \in [1, K]$ carries out mini-batch gradient descent operations locally to obtain the local gradient. Then, each participant calculates the vector Angle between the local gradient and the optimal gradient to obtain the direction evaluation of the data through the "COEs" algorithm. This work is done by the participants on the local network.

(2) Component dispersion evaluation: To calculate the participant's data dispersion rating value in the encrypted state, we performed steps 3-6 through the homomorphism of the distributed Paillier cryptographic mechanism. where the $dis(g_k^z, g_*^z)$ is the Euclidean distance between two vectors and represents the *sum* of the distance between the local gradient and the global optimal gradient of all participants.

(3) Data evaluation aggregation: After the participants obtain the evaluation value of direction and dispersion, they will conduct the evaluation combination through Eq. (3) (Step 7), and have obtained the overall evaluation of the current iteration. Here we use the product combination evaluation method, which can describe the participant data quality.

(4) Aggregated Value Update: After each participant obtains the composite evaluation, step 8 is executed to update the current optimal gradient. Each participant first computes based on the homomorphism of the distributed paillier cryptographic mechanism $\sum_{k=1}^{k=K} Q_k^Z g_k^Z$ and $\sum_{k=1}^{k=K} Q_k^z$, the specific calculation is $\sum_{k=1}^{k=K} Q_k^z g_k^z = \prod_{k=1}^{k=K} E_{p_k}(Q_k^z g_k^z)$ and $\sum_{k=1}^{k=K} Q_k^z = \prod_{k=1}^{k=K} E_{p_k}(Q_k^z)$. Then, the cloud server S performs "SAP" in collaboration with the participant to obtain the plaintext result of the *sum* of the data. The current most gradient update is then performed.

(5) weight update: The cloud server updates the global weight through Eq. (1) (Step 11), and broadcasts the global weight ω after iteration to all participants. The cloud server S and the participant repeat steps 1-11 until our preset convergence conditions are met.

Remark 5

In our setup, there are no offline users, but our scheme is highly tolerant of participants who are unable to complete the current task due to unexpected events during the execution. This is because, in the decryption process, each

participant has the privilege to perform partial decryption. Additionally, the cloud server only needs to collect partial decryption results from a minimum of t participants to obtain the decrypted result. Therefore, participants who cannot execute the scheme correctly in the current iteration are considered invalid for that iteration, but it does not affect the overall progress of the model training. In this paper, we did not consider the impact of offline participants on the model accuracy. They are treated as if they do not contribute to the current iteration's training process. However, it is important to note that in practical applications, the impact of offline participants on the model accuracy may need to be further considered and evaluated.

Correctness

Theorem 1 *Our PPFL-LQDP can accurately compute the dispersion evaluation value of participants' data in ciphertext state.*

Proof

$$E_{p_k}\left(\log\left(\frac{\sum_{k=1}^{k=K} dis}{k \cdot dis}+1\right)\right)=E_{p_k}\left(\log\left(\frac{\sum_{k=1}^{k=K} dis+k \cdot dis}{k \cdot dis}\right)\right)$$
$$=E_{p_k}\left(\log\left(\sum_{k=1}^{k=K} dis+k \cdot dis\right)\right)\cdot E_{p_k}(-\log(k \cdot dis))$$
$$=E_{p_k}(\log \mathrm{mol})\cdot E_{p_k}(-\log(k \cdot dis))$$

□

Remark 6

The discrete evaluation of participant data requires calculation in the encrypted state, and division operations pose a certain challenge to the Paillier cryptosystem. By constructing a logarithm function, the computation can be simplified without altering the distribution of evaluation values.

Security analysis

Based on Chapter 4, the threat model mainly originates from internal entities of the system, namely participants and cloud servers. Therefore, the goal of PPFL-LQDP is to ensure that the local gradients and data composite evaluation values of each participant are not leaked to other participants and cloud servers. In our constructed DCEM, the "COEs" algorithm is executed in the local network of participants, so privacy concerns are not necessary. Privacy protection is mainly focused on the "CDEs" algorithm phase and the gradient value update phase.

Due to the secure aggregation scheme "SAP" that we use, we can defend against the attack described in System model section.

- Single malicious participant attack: A single malicious participant seeks to obtain the privacy of other participants through collected information. In our "SAP" protocol, no participant knows all the private keys. Therefore, even if the malicious participant collects the encrypted gradient information sent by other participants to the cloud server, they cannot obtain the plaintext information.
- Semi-honest cloud server: Participants need to send their locally trained gradient information to the cloud server in an encrypted form, in order to offload large-scale computations to a third party. In our setup, the cloud server will perform all computations as specified, but will infer participants' information based on the knowledge acquired. During the interaction process of our scheme, the cloud server does not possess the private key, which is held by all participants in the key group. Decryption requires the collaboration of t participants. Therefore, the cloud server can only obtain aggregated information and cannot access participants' private data.
- Malicious Participants Collaborating with the Cloud Server: In the case of a collusion attack between participants and the cloud server, malicious participants can provide partial private keys to the cloud server. If the cloud server manages to collect partial private keys from t or more participants, it can access the data of all participants. Therefore, the proposed scheme can resist collusion attacks when the number of malicious participants is less than or equal to $t-1$, as distributed decryption requires collecting at least t partial decryption results before aggregating the final result. Thus, this scheme possesses a certain level of resilience against collusion attacks.

Performance analysis

In this section, we conducted training and testing tasks based on the MNIST database. This database consists of 60,000 training samples and 10,000 test samples, which are used to evaluate the proposed approach in this paper. Specifically, we compared the proposed PPFL-LQDP scheme with traditional federated learning methods and the ecProbe scheme in this paper. The experimental settings are as follows: All programs were compiled in Python and simulated on a Lenovo desktop computer with an Intel(R) Core(TM) i3-12100 3.30GHz processor and 16.0GB of RAM. The participants used Lenovo

310s laptops with an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70GHz processor for simulation. We set the learning rate to 0.01, and the batch size for gradient descent to be 128 samples per group.

Functionality

To illustrate the superiority of our PPFL-LQDP in terms of functionality, we compare it with three recent methods for privacy-preserving federated learning: SecProbe [18], PPFDL [20], and EPPFL [21]. As shown in Table 1, SecProbe, being the first framework in privacy-preserving federated learning to consider the involvement of untrusted participants, successfully defends against collusion attacks between the cloud server and users while ensuring the security of private data through differential privacy. However, if a single participant goes offline during the interaction, the normal operation of the model cannot be guaranteed. In PPFDL, improvements were made to address the offline issue of participants, but the dual-cloud architecture is challenging to ensure collusion-free in practice. As a follow-up to PPFDL, EPPFL addresses the dual-cloud issue with a single-cloud architecture and reduces the secure computation cost of scoring untrusted participants using logarithmic operations. However, in this setup, data with significant directional bias is classified as irrelevant and does not participate in training. Compared to the aforementioned methods, the PPFL-LQDP privacy-preserving framework is further built upon EPPFL. Similarly, it utilizes the distributed Paillier cryptosystem to construct a secure aggregation protocol, which addresses the issue of participants joining or going offline while ensuring privacy. Additionally, constructing DCEM and utilizing linear relationships, enables composite evaluation of the data quality of participants, allowing all participants to participate in training without losing data.

Accuracy

We discuss the accuracy of our PPFL-LQDP in this part. To illustrate the superiority of the experimental results, we compare with two representative methods, i.e., PFL [31] (Primitive Federated Learning) and SecProbe.

Remark 7

PFL is the most primitive model in federated learning, which doesn't require any operations on the data. On the other hand, SecProbe is the first privacy-preserving federated learning model that considers the participation of unreliable participants. This comparison demonstrates the necessity of handling unreliable participants in federated learning and highlights the superiority of the proposed solution. Therefore, it is reasonable to make such a comparison.

To verify the accuracy of the proposed solution after incorporating low-quality data in the training process, we randomly selected a portion of the data from the training set and introduced noise to simulate low-quality data. This same setup was applied to both the PFL and SecProbe approaches (with a privacy budget of 10 for the SecProbe solution). We conducted experiments in our Federated training using different proportions of low-quality data.

As shown in Fig. 4, we set P as the proportion of low-quality data in the training set data. The accuracy varies with P (P=10%, P=15%, P=20%, P=25%, P=30%), and the model gradually levels off with increasing number of training iterations. It can be seen that our scheme PPFL-LQDP is higher in accuracy than PFL and SecProbe, and under the same number of iterations, our scheme is also relatively accurate. At the same time, when the content of low-quality data is high, the accuracy of PFL and SecProbe decreases very fast, while our PPFL-LQDP scheme has almost no significant difference. Specific reasons:

1) PFL does not consider the addition of low-quality data, so it will cause low-quality data to over-participate in the training, directly affecting the accuracy of the model, and even making the model unusable.
2) Differential privacy to protect private data needs to be achieved by adding noise, and in order to ensure the privacy security of participants, the accuracy will cause a certain loss.

Table 1 Functional comparison with other works

	Participant privacy protection	Prevent users from going offline or joining in the middle	All the data participated in the training	Threat model	Server type
SecProbe	✓	×	✓	Semi honest	Single-Server
PPFDL	✓	×	✓	Semi honest	Dual-Servers
EPPFL	✓	✓	×	Semi honest	Single-Server
PPFL-LQDP	✓	✓	✓	Semi honest	Single-Server

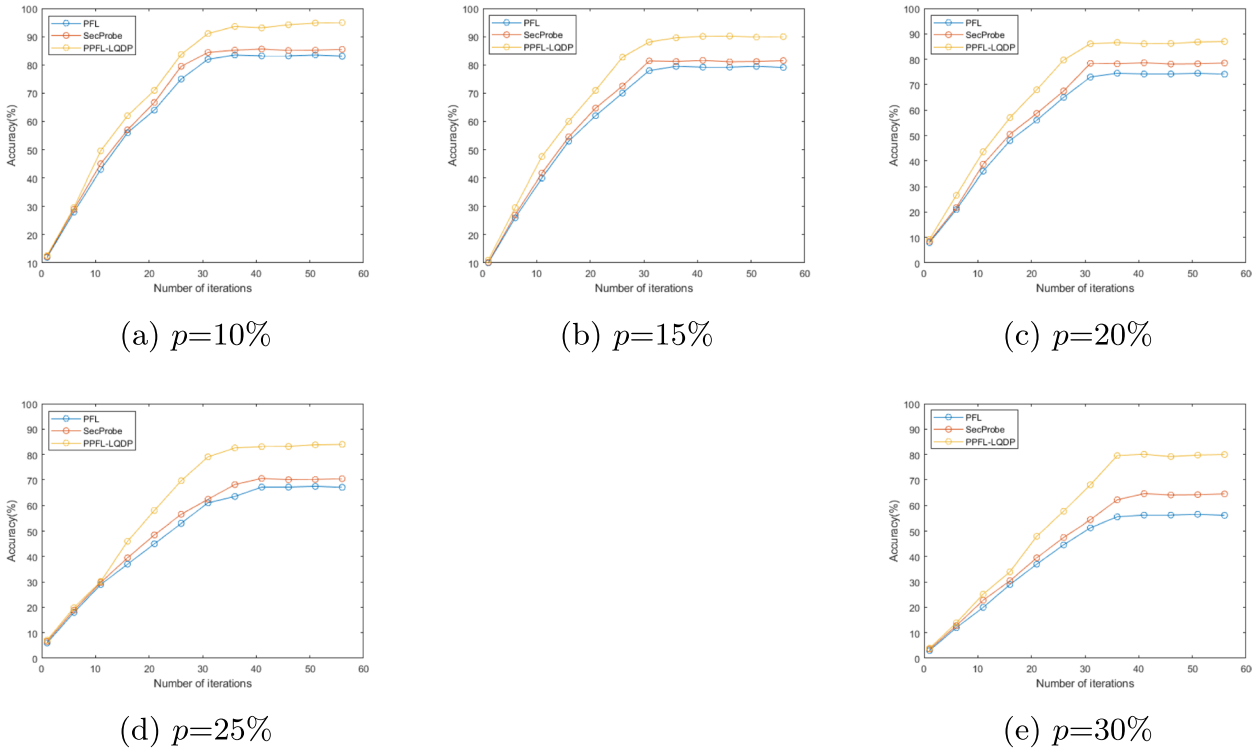


Fig. 4 Variations in model accuracy with changing proportions of low-quality data contributed by participants

However, For the low-quality data, we construct a linear relationship, so that the data can participate in the training level according to the data quality, avoiding the accuracy degradation caused by the excessive participation of low-quality data in the training process. At the same time, all the data will participate in the training, and no data loss will be caused. Referring to previous works, our scheme uses distributed paillier cryptosystem to protect private data.

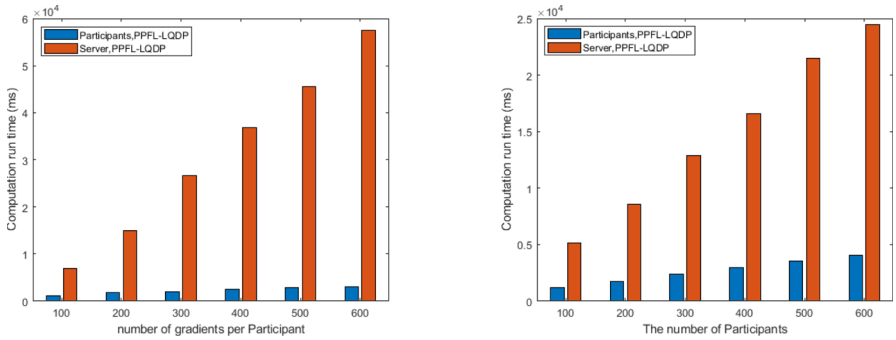
Computation overhead

In this section, we evaluated the computational overhead of the PPFL-LQDP approach. To provide a clear visualization of the experimental results, we compared them with the PPML [32] approach. To ensure the fairness of the experiment, both methods are tested using the same hardware configuration and data set. The experimental environment and learning rate are kept consistent.

The PPFL-LQDP algorithm constructed in this article utilizes distributed Paillier homomorphic encryption mechanism for additional operations to protect the privacy of participants' private data. It also performs a decryption operation to allow the cloud server to obtain intermediate values for calculating a composite evaluation value of participants' data quality.

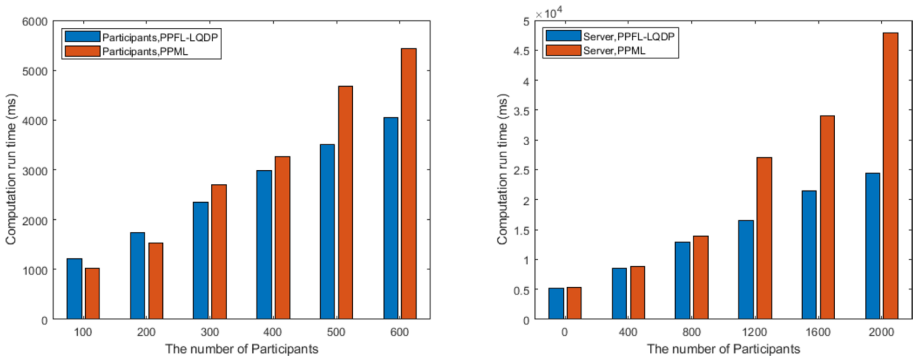
Figure 5a depicts the computational overhead of participants in the PPFL-LQDP process as the number of local gradient calculations increases. From the figure, it can be observed that the computational overhead on the participant side does not increase significantly. In PPFL-LQDP, participants only need to perform local computations of encrypted gradient sets and participate in partial decryption, while all the cryptographic operations are handled by the server. This setup is quite favorable for the participants. Furthermore, based on Fig. 5b, it can be seen that the computational cost for participants in Federated training does not increase significantly as the number of participants in the training increases.

Figure 6 illustrates the comparison of computational costs between PPFL-LQDP and PPML in terms of participant computational overhead and server computational overhead. From the figure, it is evident that as the volume of participant data increases, the computational overhead of PPML grows more rapidly compared to PPFL-LQDP. This is mainly due to the exponential increase in computational costs for PPML with an increasing number of users. However, the PPFL-LQDP algorithm can complete computations with a time complexity of $O(K + Z)$. This is primarily because the PPFL-LQDP algorithm performs composite evaluation value calculations on low-quality data on the server. The server only needs to bear the



(a) Increasing with number of gradients per participant. (b) Increasing with number of participant.

Fig. 5 Computation overhead of PPFL-LQDP



(a) The participant side. (b) The server side.

Fig. 6 Comparison of computational overhead between PPFL-LQDP and PPML

collaborative part of the "CDEs" process and the aggregation calculation of the composite evaluation values, resulting in a linear growth trend in server-side overhead. when the number of participants is large, the cost of our scheme on the server side will be better.

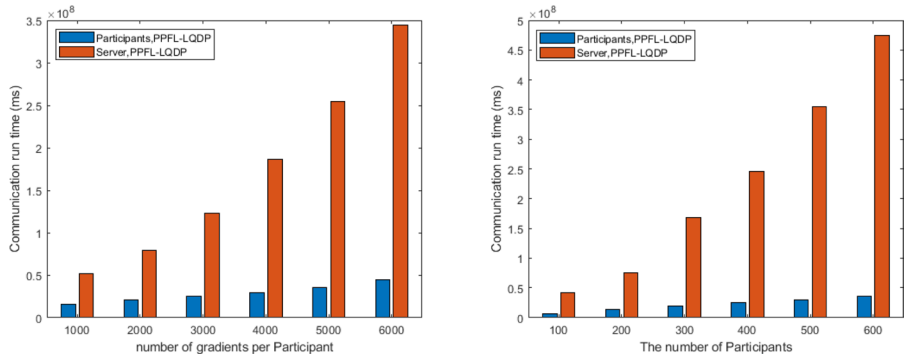
Communication overhead

Figure 7a demonstrates the variation in communication overhead between participants and the cloud server as the number of local gradient calculations increases. the majority of the communication workload is handled by the server, and the increase in overhead for each participant is not significant. In Fig. 7b, it can be observed that the communication overhead for each participant does not significantly change as the number of participants increases.

As shown in Fig. 8, the comparison of communication overhead between PPFL-LQDP and PPML can be observed. It is evident that when the number of participants varies, PPFL-LQDP incurs significantly lower communication overhead compared to PPML. This is because PPFL-LQDP only requires each participant to transmit a fixed amount of encrypted data to the cloud server, and the communication between the server and participants remains relatively stable. PPFL-LQDP also exhibits relatively stable overhead, making it participant-friendly. Therefore, compared to other approaches, PPFL-LQDP demonstrates certain advantages in terms of communication overhead.

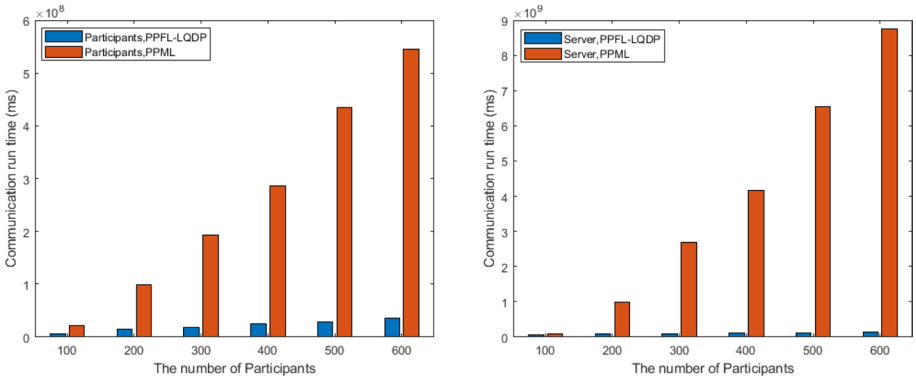
Conclusion

The paper presents a novel privacy-preserving federated learning solution, PPFL-LQDP, that addresses the issue of excessive participation of low-quality data in Federated training. By constructing a composite evaluation value for the data, the negative impact of low-quality data on



(a) Increasing with number of gradients per participant. (b) Increasing with number of participant.

Fig. 7 Communication Overhead of PPFL-LQDP



(a) The participant side. (b) The server side.

Fig. 8 Comparison of Communication Overhead between PPFL-LQDP and PPML

Federated training is reduced, while ensuring privacy and security of participant data through a secure framework. The experimental results demonstrate the capability of PPFL-LQDP in handling low-quality data, and the comparison with other approaches highlights the superior overall performance of our proposed solution.

Authors' contributions
Huiyong Wang provided the research direction and innovation points of this paper, Qi Wang was mainly responsible for the implementation of the algorithm and the writing of the code, Yong Ding, Shijie Tang and Yujue Wang mainly provided the hardware support and English polishing and all the authors participated in the writing of the manuscript.

Funding
This article is supported in part by the Guangxi Natural Science Foundatio-nunder grant 2023GXNSFAA026236, the National Natural Science Foundation of China under project 61962012, and the National Key R &D Program of China under project 2020YFB1006003.

Availability of data and materials
The datasets are available online.The URL is as follows: MNIST database: http://yann.lecun.com/exdb/mnist.

Declarations

Ethics approval and consent to participate
This declaration is not applicable.

Competing interests
The authors declare no competing interests.

Received: 7 October 2023 Accepted: 24 February 2024
Published online: 18 March 2024

References

1. Jordan MI, Mitchell TM (2015) Machine learning: Trends, perspectives, and prospects. Science 349(6245):255–260
2. Fleming N (2018) How artificial intelligence is changing drug discovery. Nature 557(7706):S55–S55
3. Zhou ZH (2016) Learnware: on the future of machine learning. Front Comput Sci 10(4):589–590
4. Liu B, Ding M, Shaham S, Rahayu W, Farokhi F, Lin Z (2021) When machine learning meets privacy: A survey and outlook. ACM Comput Surv (CSUR) 54(2):1–36
5. De Cristofaro E (2021) A critical overview of privacy in machine learning. IEEE Secur Priv 19(4):19–27

6. McMahan HB, Moore E, Ramage D, y Arcas BA (2016) Federated learning of deep networks using model averaging. arXiv preprint arXiv:1602.05629
7. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, pp 1273–1282
8. Aono Y, Hayashi T, Wang L, Moriai S (2017) Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forensic Secur 13(5):1333–1345
9. Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the GAN: Information leakage from collaborative deep learning. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Dallas, 30 October-3 November 2017. https://doi.org/10.1145/3133956.3134012. pp 603–618
10. Richardson A, Filos-Ratsikas A, Faltings B (2020) Budget-bounded incentives for federated learning. Federated Learn Priv Incent vol.12500:176–188
11. Liang W, Tadesse GA, Ho D, Fei-Fei L, Zaharia M, Zhang C, Zou J (2022) Advances, challenges and opportunities in creating data for trustworthy AI. Nat Mach Intell 4(8):669–677
12. Fouque P A, Poupard G, Stern J (2000) Sharing decryption in the context of voting or lotteries. In: Financial Cryptography: 4th International Conference, Anguilla, British West Indies, 20-24 February 2000. pp 90–104
13. Wei K, Li J, Ding M, Ma C, Yang HH, Farokhi F, Poor HV (2020) federated learning with differential privacy: Algorithms and performance analysis. IEEE Trans Inf Forensic Secur 15:3454–3469
14. Falcetta A, Roveri M (2022) Privacy-preserving deep learning with homomorphic encryption: An introduction. IEEE Comput Intell Mag 17(3):14–25
15. Ma C, Li J, Ding M, Yang HH, Shu F, Quek TQ, Poor HV (2020) On safeguarding privacy and security in the framework of federated learning. IEEE Netw 34(4):242–248
16. Fang H, Qian Q (2021) Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning. Future Internet 13(4):94
17. Park J, Lim H (2022) Privacy-preserving federated learning using homomorphic encryption. Appl Sci 12(2):734
18. Zhao L, Wang Q, Zou Q, Zhang Y, Chen Y (2019) Privacy-preserving collaborative deep learning with unreliable participants. IEEE Trans Inf Forensic Secur 15:1486–1500
19. Yu H, Liu Z, Liu Y, Chen T, Yang Q (2020) A Sustainable Incentive Scheme for Federated Learning. IEEE Intell Syst 35(4):58–69
20. Xu G, Li H, Zhang Y, Xu S, Ning J, Deng RH (2020) Privacy-preserving Federated deep learning with irregular users. IEEE Trans Dependable Secure Comput 19(2):1364–1381
21. Li Y, Li H, Xu G, Huang X, Lu R (2021) Efficient privacy-preserving federated learning with unreliable users. IEEE Internet Things J 9(13):11590–11603
22. Nguyen DC, Ding M, Pathirana PN, Seneviratne A, Li J, Poor HV (2021) federated learning for internet of things: A comprehensive survey. IEEE Commun Surv Tutorials 23(3):1622–1658
23. Danner G, Jelasity M (2015) Fully distributed privacy preserving mini-batch gradient descent learning. In: Distributed Applications and Interoperable Systems: 15th IFIP WG 6.1 International Conference, Grenoble, France, 2-4 June 2015, Springer, Cham. pp 30–44
24. Damgård I, Jurik M (2001) A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, 13-15 February 2001 Proceedings 4. Springer Berlin Heidelberg. pp 119–136
25. Bohli JM, Gruschka N, Jensen M, Iacono LL, Marnau N (2014) Security and privacy-enhancing multicloud architectures. IEEE Trans Dependable Secure Comput 10(4):212–224
26. Xu G, Li H, Lu R (2018) Practical and privacy-aware truth discovery in mobile crowd sensing systems. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. https://doi.org/10.1145/3243734.3278529. pp 2312–2314
27. Choi I, Song Q, Sun K (2019) Federated-cloud based deep neural networks with privacy preserving image filtering techniques. In: IEEE Conference on Dependable and Secure Computing (DSC), Hangzhou, China, November 2019. https://doi.org/10.1109/DSC47296.2019.8937635. pp 1–8
28. Vaziri R, Mohsenzadeh M, Habibi J (2019) Measuring data quality with weighted metrics. Total Qual Manag Bus Excell 30(5–6):708–720
29. Díaz C, Calderon-Ramirez S, Aguilar LDM (2022) Data Quality Metrics for Unlabelled Datasets. In: IEEE 4th International Conference on BioInspired Processing (BIP), Cartago, Costa Rica, November 2022. https://doi.org/10.1109/BIP56202.2022.10032475. pp 1–7
30. Luping W, Wei W, Bo L C (2019)CMEL: Mitigating Communication Overhead for Federated Learning. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Dallas, TX, USA. https://doi.org/10.1109/ICDCS.2019.00099. pp 954–964
31. Smith V, Chiang CK, Sanjabi M, Talwalkar AS (2017) Federated multi-task learning. Adv Neural Inf Process Syst 30:4424–4434
32. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: proceedings of the ACM SIGSAC Conference on Computer and Communications Security, New York, USA, October 2017. https://doi.org/10.1145/3133956.3133982. pp 1175–1191

Publisher's Note