ADP-PFL: Differential Privacy Federated Learning Based on Privacy Budget Allocation and Sparsification

No Author Given

No Institute Given

Abstract. Federated learning is a distributed machine learning method that enables multiple clients to train a model collaboratively. Since the data remains stored locally, this method effectively protects the privacy of the clients' information. Nonetheless, in today's context, federated learning still faces risks of privacy leakage. Differential privacy techniques are widely used in federated learning to safeguard client privacy, where the size of the privacy budget directly affects the model's utility. The present research has not fully considered varying privacy needs, as client parameters in different communication rounds pose varying privacy leakage risks. To address these challenges, we propose a differential privacy federated learning algorithm based on privacy budget allocation and sparsification, named ADP-PFL. This approach introduces an adaptive privacy budget allocation mechanism, which dynamically allocates privacy budgets to clients in different communication rounds based on quantified privacy leakage risks, providing adaptive privacy protection. Simultaneously, model pruning reduces the dimensionality of parameters to decrease sensitivity, effectively reducing the amount of noise added after applying differential privacy. Experimental results show that our algorithm outperforms existing state-of-the-art algorithms in accuracy, effectively balancing privacy protection and model utility.

Keywords: Federated Learning · User-level Differential Privacy · Privacy Budget Allocation.

1 INTRODUCTION

With the advent of the big data era, data collection and analysis are increasingly important in various fields. However, privacy protection issues have also become prominent, especially in personal privacy data scenarios. Federated learning [1] is a machine learning paradigm that allows multiple clients to train a model collaboratively without sharing data. In this process, clients only transmit updated model parameters to a server for aggregation, thus constructing a global model.

Moreover, privacy risks still exist in federated learning. Although local data in federated learning is not directly shared, attackers can indirectly leak clients' privacy from the transmitted model parameters or gradients through inference attacks [2] or model inversion attacks [3], among other methods. Current research adopts technologies such as homomorphic encryption [4], secure multi-party computation [5], and differential privacy [6] to solve privacy protection issues in federated learning. However, while homomorphic encryption and secure multi-party computation provide high security, they often come with significant computational overhead. In contrast, differential privacy can protect privacy while reducing computational overhead.

Many studies are dedicated to solving privacy leakage issues in federated learning using differential privacy technology. In federated learning with user-level differential privacy [7], it is necessary to clip thresholds to control sensitivity and add Gaussian noise to the aggregated gradients to protect client privacy. The earliest studies [8], [9] set a fixed privacy budget ϵ and added constant noise to the parameters, affecting the model's accuracy and convergence performance. Therefore, recent research has started designing adaptive privacy protection methods. However, literature [10] and [11] only consider adaptive noise perturbation during the client's local training process without providing adaptive privacy protection for different communication rounds. Literature [12] uses accuracy to allocate privacy budgets. Still, accuracy cannot reflect the privacy leakage risk of parameters, and relying on it for privacy budget allocation cannot better adapt to different rounds of privacy budgets. Moreover, literature [13], [14] also applies sparsification technology to federated learning privacy protection. These two algorithms are applied in federated learning with user-level differential privacy and use sparsification to reduce communication costs. Conversely, our work uses sparsification technology to improve the model utility after using differential privacy.

To address these challenges, we propose the ADP-PFL. This adaptive privacy protection federated learning framework provides personalized privacy protection for clients in different communication rounds while

ensuring the training performance of the federated learning model. Specifically, we propose an adaptive privacy budget allocation mechanism that dynamically allocates privacy budgets to clients in different communication rounds based on the privacy leakage risk quantified through the Jacobian matrix, providing adaptive privacy protection. Additionally, to improve the training performance of the federated learning model, we propose an adaptive model sparsification mechanism, which uses model pruning to reduce the dimensionality of parameters, thereby lowering sensitivity and effectively reducing the amount of noise added after applying differential privacy, balancing privacy and usability. The main contributions of this paper are summarized as follows:

1. Propose an adaptive privacy budget allocation algorithm based on privacy leakage risk. This algorithm quantifies information leakage risk by calculating the impact of input data on model output through the Jacobian matrix. At the same time, the neural network training process adaptively allocates different privacy budgets for different rounds, avoiding rapid consumption of the privacy budget.

2. Calculating the Fisher information matrix of the model eliminates some parameter updates that have a minor impact on model performance to reduce the norm of local updates further, improving communication efficiency while reducing the noise amount added by differential privacy and achieving a balance between privacy and utility.

3. We evaluated the proposed algorithm on multiple datasets, and the experimental results show that the ADP-PFL algorithm outperforms existing algorithms in model accuracy, model loss, and other aspects.

The remainder of the article is organized as follows: Section 2 reviews related work on personalized federated learning and differential privacy federated learning; Section 3 introduces the basics of federated learning and differential privacy; Section 4 details the proposed ADP-PFL algorithm and its proof; Section 5 presents the experimental results; finally, Section 6 summarizes the paper.

2 RELATED WORK

2.1 User-Level Differential Privacy in Federated Learning

The concept of user-level differential privacy in federated learning was introduced by McMahan et al. [7]. They proposed the DP-FedAvg algorithm to train mobile keyboard prediction models, ensuring user-level differential privacy through the Gaussian mechanism and privacy accounting via moment accountant. Wei et al. [15] perturb model parameters using the Gaussian mechanism before sending them to the server for aggregation, appropriately adjusting the noise variance and theoretically analyzing the convergence bounds of the privacy protection level. Zhang et al. [16] proposed training a personalized federated model optimization framework with DP-SGD, a centralized variant of differential privacy that adds a tunable amount of Gaussian noise to each gradient, thereby reducing the link between model updates and individual samples in local training data. Geyer et al. [17] introduced a user-level differential privacy federated learning framework that conceals users' contributions during model training and balances between privacy loss and model performance. Cheng et al. [18] built upon DP-FedAvg, introducing bounded local update regularization and local update sparsity. They first introduced a regularization term into the local optimization function and limited the L2 norm of local updates within a certain range. Before clipping, they zero out some updates that have little impact on local model performance, thus reducing the norm of local updates without compromising local model accuracy. Shi et al. [19] proposed a differential privacy federated learning algorithm named DP-FedSAM. This algorithm mitigates the negative impact of differential privacy through gradient perturbation. It integrates the Sharpness Aware Minimization optimizer, generating local flat models with better stability and robustness to weight perturbations. It enhances the local model's robustness to differential privacy noise, improving performance. However, these methods do not fully address the impact of a fixed privacy budget on model performance and use a uniform noise addition strategy, making it difficult to meet the varying privacy needs of different users.

2.2 Adaptive Privacy Budget Allocation

In federated learning, privacy protection can be achieved by injecting differential privacy noise into the model. However, a reasonable allocation of the privacy budget can improve the model's accuracy and reduce the impact of model noise. Huang et al. [20] proposed a new differential privacy federated learning framework,

DP-FL, which adaptively adds noise to the client side. Unlike traditional privacy budget allocation methods, this is divided into two parts: one is used for adding noise to the gradients, and the other is used for selecting the optimal step size. Dong et al. [21] proposed a personalized and adaptive differential privacy federated meta-learning algorithm, PADP-FedMeta, which allocates different privacy budgets according to the distribution of client sample labels. It also sets a set of selectable learning rates locally, and if the optimal learning rate is found, parameters are updated; otherwise, the value of the privacy budget is increased. Yang et al. [22] proposed an algorithm, PLU-FedOA, where PLU allows clients to update parameters locally and select personalized privacy budgets. FedOA helps the server aggregate optimized local parameters. Shen et al. [23] introduced a federated learning scheme based on personalized local differential privacy by adjusting privacy budget parameters and introducing different security range parameters for each client to achieve personalized privacy protection. Yang et al. [24] proposed a gradient compression federated learning framework, GFL-ALDPA, based on adaptive localized differential privacy budget allocation to reduce the loss of privacy budgets and model noise. By allocating different privacy budgets in different communication rounds during the training process, the limited privacy budget can be maximized, improving model accuracy. Although differential privacy technology as a security mechanism for federated learning has been extensively studied, the trade-off between model performance and privacy protection levels remains a major issue in protecting federated learning with differential privacy technology.

3 BASIC TECHNOLOGY

3.1 Federated Learning

The process of federated learning is defined as follows: Let D_i represent the local data held by the i-th client, where $i \in \{1, 2, \ldots, N\}$. The server aims to learn a global model from the data of N clients and minimize the model's loss function. Formally, the server aggregates the model weights received from the N clients as:

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

where w_i are the model parameters trained by the i-th client, w are the model parameters aggregated by the server, N is the number of clients, $p_i = \frac{|D_i|}{|D|} \geq 0$ and $\sum_{i=1}^{N} p_i = 1$, $|D| = \sum_{i=1}^{N} |D_i|$ is the total number of all data samples. This optimization problem can be expressed as:

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

In the equation above, $F_i(w, D_i)$ is the local loss function of the i-th client. In federated learning, N clients collaborate under the server to train the model jointly, ultimately optimizing (2) to converge to the global optimum. To address this optimization task, federated learning employs multiple rounds of communication between the local training and server aggregation phases, as shown in Fig. 1. In local training, clients initialize and train the global model locally and then upload the updates. The server aggregates these models to update and redistribute the global model.

3.2 Differential privacy

Differential privacy is a rigorous mathematical mechanism that formally defines privacy loss. It stipulates that any change in the training dataset should not cause significant variations in the algorithm output, thereby achieving the objective of privacy protection.

Definition 1 (Differential Privacy [6]): Let $M : D \to R$ be a random mechanism, D and D' are neighbouring datasets that differ by at most one record. If for any output $S \subseteq R$ the random mechanism M satisfies the following equation on both D and D', then the mechanism M satisfies (ε, δ)-DP:

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

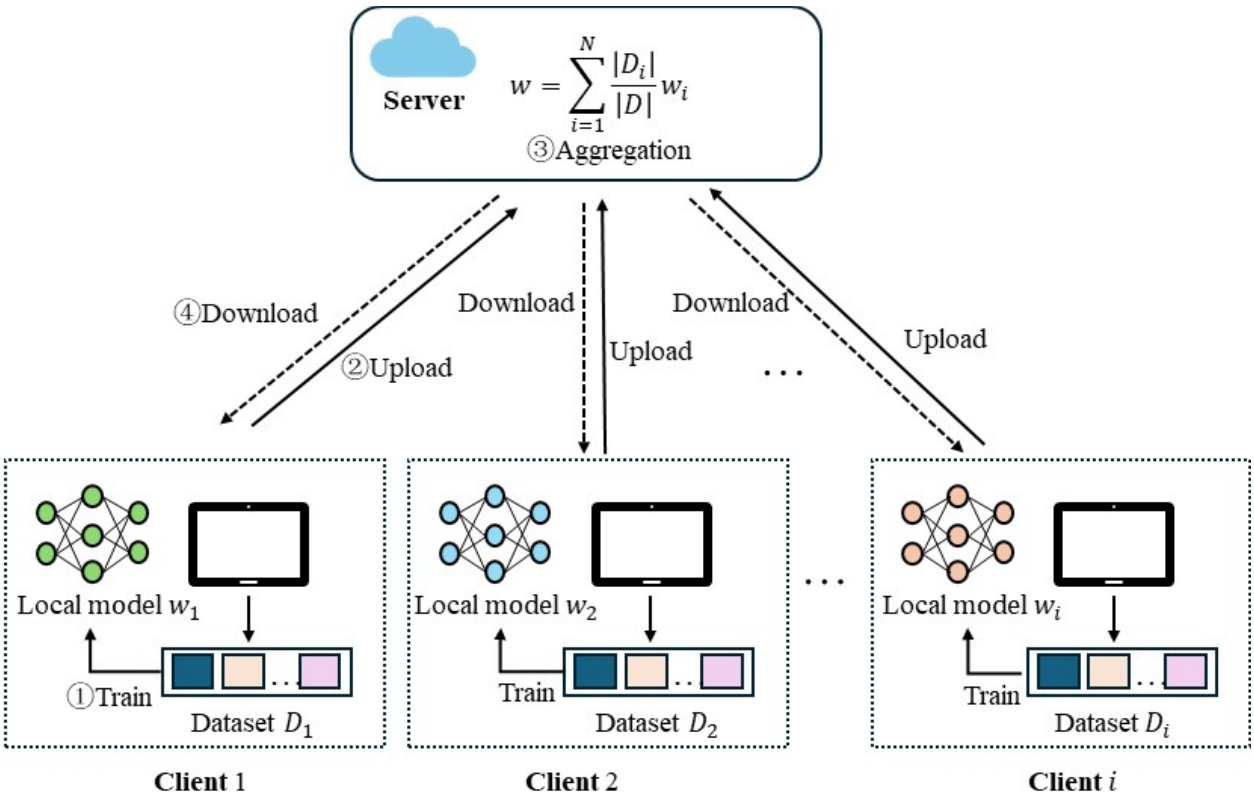


Fig. 1: Federated Learning

The parameter ε represents the privacy budget, reflecting the degree of privacy protection the algorithm offers. A smaller ε indicates a higher level of privacy protection. δ is a relaxation term, representing the probability of deviating from pure ε-DP.

Definition 2 (Rényi Divergence [25]): Given two probability distributions P and Q, the Rényi-α divergence between P and Q is defined as:

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log E_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)}\right)^\alpha\right] \tag{4}$$

Definition 3 (Rényi Differential Privacy [25]): For any two adjacent datasets D and D', if there exists a random mechanism $M : D \to R$ that satisfies the following equation, then the random mechanism M is said to satisfy (α, ε)-RDP:

$$D_\alpha(M(D) \parallel M(D')) \leq \varepsilon \tag{5}$$

where, $D_\alpha(M(D) \parallel M(D')) \leq \varepsilon$ indicates that the Rényi divergence between the distributions on adjacent datasets is limited within the privacy parameter ε. As $\alpha \to \infty$, (α, ε)-RDP converges to pure differential privacy.

3.3 Jacobian matrix

The Jacobian matrix [26] is a mathematical expression used to represent all the first-order partial derivatives of a vector-valued function concerning a set of variables. Each matrix element is a partial derivative, expressing a function component's local rate of change concerning a specific variable.

The Jacobian matrix is very important in mathematics, providing the best linear approximation of a function at a given point. For a function $f : R^n \to R^m$, where the input is $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and the output is $\mathbf{y} = (y_1, y_2, \ldots, y_m)$, the Jacobian matrix is the matrix of all first-order partial derivatives of

the vector-valued function. This matrix is an $m \times n$ matrix, and each element of the matrix is the partial derivative of one component of the function concerning one of the variables, denoted as $\frac{\partial y_i}{\partial x_j}$.

The Jacobian matrix is important because it provides the best linear approximation of a function near a given point. In fields such as engineering, physics, and computer science, it is widely used to solve problems in nonlinear systems, such as optimization, control theory, robotics, and kinematics.

3.4 Fisher information

Fisher information [27] is a very significant concept in statistics, and it plays an essential role in parameter estimation and hypothesis testing. This concept is derived from the likelihood function of statistical models. Generally, the larger the Fisher information of a parameter, the more information about the parameter can be obtained from the sample data, and this indicates a higher precision in parameter estimation. Fisher information measures the amount of information an observable random variable carries about an unknown parameter upon which the likelihood is based. Simply put, it quantifies the steepness or sharpness of the likelihood function concerning the parameter, thus reflecting the model's sensitivity to the parameter.

Fisher information is an important concept in the optimal design of experiments and forms the basis for many statistical methods. For a given model, the Fisher information for a single parameter is defined as:

$$I(\theta) = -E\left[\frac{\partial^2}{\partial \theta^2} \log L(\theta)\right] \tag{6}$$

where $L(\theta)$ is the likelihood function, and $-E[]$ represents the expectation. In parameter estimation, Fisher information provides a method to assess the efficiency of an estimator. According to the Cramér-Rao bound, under certain regular conditions, Fisher information provides a lower bound on the variance of any unbiased estimator, thereby limiting the precision of parameter estimation. In other words, the greater the Fisher information for a parameter, the smaller the lower bound for the variance of an unbiased estimator, which implies a potentially more accurate estimation.

4 METHOD

In this paper, we introduce an Adaptive Privacy-Preserving Federated Learning (ADP-PFL) framework to address privacy leakage issues in federated learning, as illustrated in Fig. 2. Initially, the ADP-PFL framework incorporates an adaptive privacy budget allocation mechanism, which dynamically adjusts the distribution of privacy budgets based on the privacy leakage risk among clients and the number of training rounds. This adaptive mechanism is primarily implemented by calculating the Jacobian matrix, which quantifies the sensitivity of model outputs to input changes. Based on this metric, our framework can identify the model parameters and training stages crucial for privacy protection, allocating appropriate privacy budgets in different communication rounds. Additionally, the ADP-PFL framework calculates the Fisher information matrix to determine the importance of each parameter during the model training process. This allows the framework to recognize and retain parameter updates that significantly impact model performance while eliminating those with little effect. This method not only enhances the communication efficiency of the model but also achieves a balance between privacy protection and model performance by reducing the dimensionality of parameters, thus minimizing the noise perturbation associated with differential privacy.

4.1 Privacy Budget Allocation

Quantifying Leakage Risk Existing research indicates [28] that information leakage is usually caused by insufficient generalization performance or inadvertent memorization. Understanding the memorization of private information by models is closely related to their generalization abilities: models with high generalization capabilities can avoid memorizing unnecessary information, making their parameters less sensitive to minor changes in the input. If certain model parameters are insensitive to changes in the input data, the difficulty of reconstruction attacks increases, reducing the success rate of such attacks. The sensitivity of input data is an indicator that measures the extent to which minor changes in the input data affect the model's output and can be used to evaluate the model's generalization performance [29]. Therefore, this paper uses the

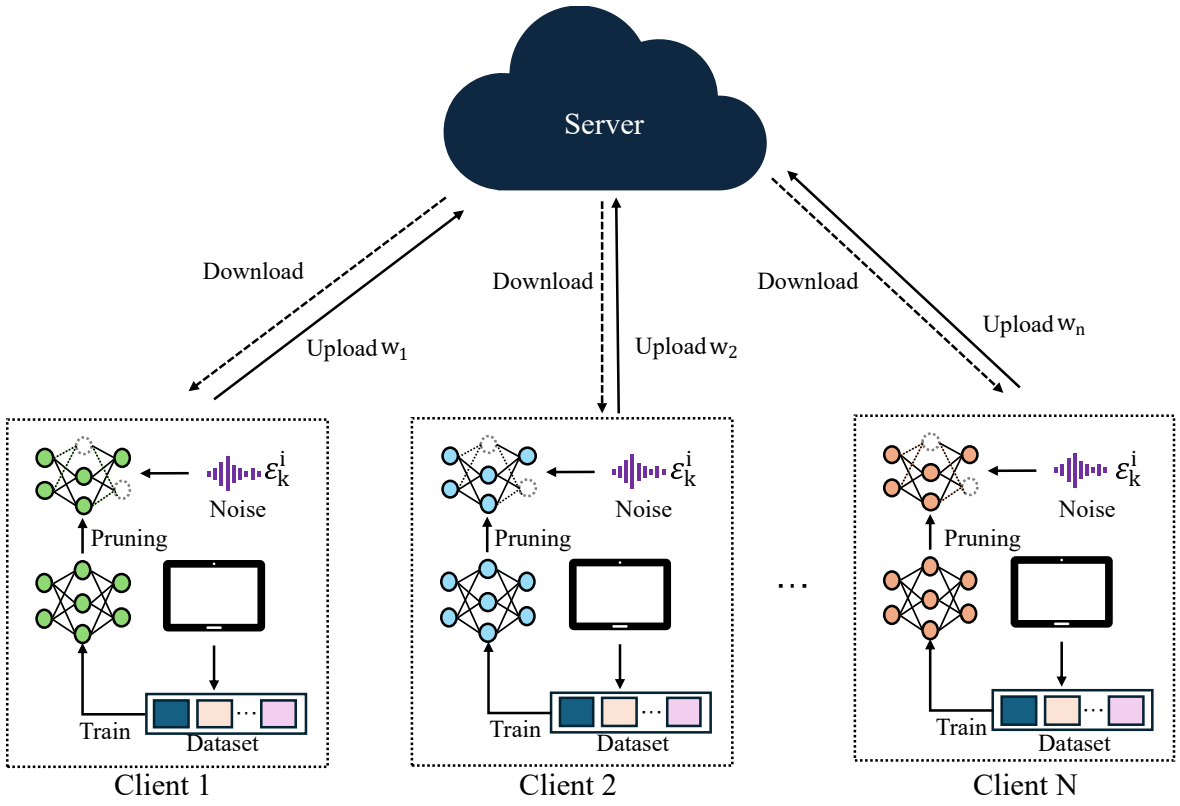


Fig. 2: ADP-PFL framework

Jacobian matrix to measure the model's generalization performance and, from there, to assess the privacy risk. To begin with, the Jacobian matrix of the model relative to the input sample X, where X refers to the training dataset of the client, is calculated. The dependency of model parameters on input data is quantified through the Jacobian matrix, providing a method to assess the model's sensitivity to the input data. The formula for calculating the Jacobian matrix of model parameters concerning the input is:

$$p^w(X) = \frac{\partial g(X)}{\partial X} = \frac{\partial}{\partial X}\left[\frac{\partial e(X, y, \theta)}{\partial \theta}\right] \tag{7}$$

The $p^w(X)$ represents the weight-sensitive part of the model's output concerning the input X. It can analyze the model's sensitivity to the input data. The smaller the sensitivity, the less likely the memorization of unnecessary details in the input data, which is conducive to the protection of privacy. It can be calculated as follows.

In addition, for the k-th output of the model, the p-norm can be used to measure the sensitivity of the model to the input data, which can be calculated as follows for the input X:

$$R^X = \frac{1}{k}\sum_{i=1}^{k} \|p^w(X)\|_p \tag{8}$$

The smaller the value of R^X, the smaller the sensitivity of the model, or in other words, the better the generalization ability of the model and the lower the risk of privacy leakage.

Privacy Budget Allocation In the training process of federated learning, the reasonable allocation of a privacy budget is crucial to the model's learning. Significant changes to the model parameters initially allow for larger noise in the updates without causing the model to deviate noticeably from the correct learning

path. However, as training progresses and the magnitude of parameter adjustments decreases, even minor noise can adversely affect performance. Therefore, controlling the privacy budget at the beginning of training is important and then gradually increasing it for each training round as the model stabilizes. This approach not only protects privacy but also saves on the privacy budget. Moreover, this paper notes that, in different communication rounds of federated learning, the degree of attack threats faced by different clients varies. Uniformly allocating the privacy budget often provides excessive or insufficient privacy protection, resulting in decreased model accuracy and convergence speed or compromised client privacy. Based on the analysis above, this paper designs a method that gradually increases the privacy budget with the number of training rounds and allocates a personalized budget for each client based on their specific privacy risks, thereby protecting privacy without sacrificing model performance.

Algorithm 1 Privacy Budget Allocation

Input: local model w_i; clipping threshold C; privacy budgets ϵ; number of epochs E
Output: global model
//Client side
for $t = 1, 2, \ldots, T$ **do**
  Select a subset of clients randomly
  for each selected client i **do**
    $w_i^t \leftarrow$ Download w^{t-1}
    for each local epoch j from 1 to E **do**
      Sample batch $B \subseteq D_i$
      for each batch $b \in B$ **do**
        $w_i^t \leftarrow w_i^t - \eta \nabla L(w_i, b)$
      end for
    end for
    Quantify privacy risks $P^w(X)$ using Jacobian matrix
    $P^w(X) = \frac{\partial g(X)}{\partial X} = \frac{\partial}{\partial X}\left[\frac{\partial \theta(X,y,\theta)}{\partial \theta}\right]$
    $\hat{w}_i^t \leftarrow$ model sparsification
    $\hat{w}_i^t \leftarrow \hat{w}_i^t / \max(1, \frac{\|\hat{w}_i^t\|_2}{\zeta})$
    $\tilde{w}_i^t = \hat{w}_i + N(0, C^2\sigma^2)$
  end for
end for
//Server side
Round-level budget allocation:
$\epsilon^t = \exp(a) \cdot \epsilon_r \cdot \frac{t}{T-t+1}$
Client-level budget allocation:
$\epsilon_i^t \leftarrow \epsilon \cdot \frac{\min(R_1, \ldots, R_n)}{R_i^{t-1}}$
Model aggregation:
$w^{t+1} \leftarrow \sum_{i=1}^{S} \frac{|D_i^t|}{|D|} \tilde{w}_i^t$
return

First, according to the sequential composition theorem, the sum of the privacy budgets for all training rounds must not exceed the preset total budget ε. Specifically, the privacy budget ceiling for each round is determined by the following formula:

$$\epsilon^t \leq \frac{\epsilon_r}{T - t + 1} \tag{9}$$

The remaining privacy budget ϵ_r is given by $\epsilon_r = \epsilon - \epsilon_c$, where ϵ is the total privacy budget, ϵ_c is the consumed privacy budget, and ϵ_r is the privacy budget that can be allocated in the future. ϵ_c is the sum of the privacy budgets consumed in the previous rounds, and t is the number of the current round.

$$\alpha = -\beta \left(1 - \left(\frac{t}{T}\right)^2\right) \tag{10}$$

$$\epsilon^t = \exp(\alpha) \times \epsilon_r \times \frac{t}{T-t+1} \tag{11}$$

where (11) on the right side corresponds to the scenario where the remaining privacy budget is uniformly distributed in subsequent rounds. β is a hyperparameter that adjusts the decay rate. $\frac{t}{T}^2$ will increase faster with the number of training rounds, and accordingly, $1 - \left(\frac{t}{T}\right)^2$ will decrease overall more significantly, achieving a slow decline initially and a rapid decline later. This formula ensures that, as the training rounds progress, the privacy budget allocated increases gradually to accommodate the training of the model.

Secondly, given the privacy budget ϵ_t for the current round, to meet the personalized privacy needs of clients, the server selects an appropriate privacy budget $\epsilon_{i,t}$ for each client based on the measured privacy leakage risk of each client as the privacy budget for that client in this round.

$$\epsilon_i^t = \epsilon^t \left(\frac{\min(R_1, \ldots, R_n)}{R_i^{t-1}} \right) \tag{12}$$

where, $\min(S_1, \ldots, S_n)$ represents the minimum Jacobian norm among all clients, ϵ_t is the privacy budget for the t-th round, and $\epsilon_{i,t}$ is the privacy budget for the n-th client in the t-th round. Notably, in the first round, this paper does not allocate client-level privacy budgets, but in subsequent rounds, privacy budgets are allocated based on the historical privacy leakage risks of the clients. Formula (12) ensures that clients with higher privacy risks are allocated lower privacy budgets, while those with lower risks receive higher budgets, achieving a balance between privacy protection and model utility.

In Algorithm 1, clients are randomly selected (line 2) to download the latest global model weights w^{t-1} (line 4) and proceed with local training over E epochs (line 5), applying a clipping threshold C to ensure gradient stability (line 8). During these epochs, privacy risks are quantified using a Jacobian matrix (line 11), which informs the subsequent sparsification and normalization of weights (lines 13-14), followed by noise addition for enhanced privacy (line 15). On the server side, dynamic management of the privacy budget is performed using an exponential decay function (line 18), followed by client-level budget allocation (line 20) and model aggregation from all client updates into a new global model w^{t+1} (line 23). This comprehensive process aims to optimize the global model's effectiveness while adhering to stringent privacy standards. It balances performance and data security as it cycles through adjustments and evaluations until convergence or the desired accuracy is achieved (line 24).

4.2 Model Sparsification

The Gaussian mechanism requires model parameters to be clipped to a clipping threshold C and noise proportional to C to be added during model updates. A small clipping threshold can lead to significant bias, while a large threshold can cause increased variance, slowing down the convergence and impairing the performance of the global model. The motivation of this paper is to naturally reduce the L_2 norm of the local model before clipping, thus making parameter clipping more adaptable to the local model parameters. Furthermore, reducing the model norm can further decrease sensitivity and reduce the addition of noise. Fisher information can quantify the amount of information that parameters can provide; parameters with greater information content can represent knowledge more effectively and play a crucial role in model prediction. Therefore, before clipping, this paper will measure the Fisher information of each parameter, zeroing out those with lower Fisher information, i.e., parameters that have less impact on local model performance, thereby reducing the norm of the local model without compromising its accuracy.

We calculate the Fisher information $F_i \in R^{d_w}$ for each parameter, where d_w represents the dimension of the model parameters. This measure provides the element for w_i in the diagonal matrix of Fisher information contributed by the gradient of the client:

$$F(w_{ij}) = \left(\frac{\partial \log L(w_i, D_i)}{\partial w_{ij}} \right)^2 \tag{13}$$

where $\log L(w_i, D_i)$ represents the log-likelihood function of w_i.

Next, we normalize the Fisher information of the parameters for each client to compute the Fisher information for each parameter:

$$\hat{F}_{k,j} = \frac{F_{k,j} - \min\{F_{k,j}\}}{\max\{F_{k,j}\} - \min\{F_{k,j}\}} \tag{14}$$

Based on the normalized Fisher information of each parameter, a mask matrix M is generated to perform parameter clipping. On each client M_i, if the Fisher information of a parameter is greater than a certain threshold τ, then that parameter is kept as 1; otherwise, it is set to 0. The method depends on determining the threshold of Fisher information for the model update gradient.

$$M[j] = \begin{cases} 1, & \text{if } \hat{F}_{k,j} > \tau \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Then, this paper performs a Hadamard product between the parameters before clipping to select the retained parameters. Specifically, parameters with higher Fisher information, representing important information in the model, are kept, while others are set to 0. This operation can reduce the L_2 norm of the parameters without compromising model performance.

Afterwards, we perform a Hadamard product between the mask matrix M and the parameters before clipping to select the parameters to retain. In this case, the parameters with higher Fisher information, which are more important to the model, are kept as 1 in the matrix M, and the less important ones are set to 0. This process is described as follows:

$$\hat{w}_i^t = M \odot w_i \tag{16}$$

where perform a Hadamard product on M and w_i to obtain the clipped weights $\hat{w}_i$.

Algorithm 2 Model Sparsification

1: **Input:** rounds T, number of clients N, local dataset D_i, threshold τ
2: **Output:** sparse local model w_i^t
3: **for** each round $t = 1, 2, \ldots, T$ **do**
4: **for** each client i **do**
5: Compute Fisher Information $F(w_{ij})$
6: $F(w_{ij}) = \left(\frac{\partial \log L(w_{ij}, D_i)}{\partial w_{ij}}\right)^2$
7: Normalize Fisher Information
8: $F_{k,j} = \frac{F_{k,j} - \min(F_{k,j})}{\max(F_{k,j}) - \min(F_{k,j})}$
9: Compute mask matrix M
10: $M[j] = \begin{cases} 1, & \text{if } F_{k,j} > \tau \\ 0, & \text{otherwise} \end{cases}$
11: $\hat{w}_i^t \leftarrow M \odot w_i^t$
12: **end for**
13: **end for**
14: **return** $\hat{w}_i^t$

In Algorithm 2, the iterative process initializes model weights w^0 and updates through multiple rounds and clients. Fisher Information $F(w_{ij})$ for each weight is calculated to guide sparsification, assessing each weight's importance by squaring the derivative of the log-likelihood (lines 3-5). This information is normalized and used to construct a mask matrix M, setting values based on a threshold τ to determine significance (lines 7-9). Weights below the threshold are zeroed out, creating a sparser model w_i^t (line 10). The process cycles through adjustments and performance testing until convergence or desired accuracy is achieved, and the refined sparse model is finalized (line 14).

The norm of the sparsified $\|\hat{W}\|$ will always be less than $\|W\|$. By adjusting τ, one can control the sparsity of the local model, thereby adjusting the model's norm before the pruning operation. Due to the reduction in the model norm, as defined by differential privacy, the sensitivity of the model decreases, leading to a reduction in the added noise. The specific proof is as follows:

Theorem 1. *In federated learning, if we assume that the values of each dimension of the parameters follow the same distribution, then the application of a sparsification mechanism reduces the real sensitivity from Δf to $\Delta f(1-p)$, where $p \in (0,1)$ represents the degree of sparsification.*

Proof. Assume $f_d \geq f'_d$, and $f_d = (y_1, y_2, \ldots, y_k)$, $f'_d = (y'_1, y'_2, \ldots, y'_k)$, where d denotes the dimension of the parameters. The calculation formula for the L_2 sensitivity can be written as follows:

$$\Delta f = \sqrt{(y_1 - y'_1)^2 + (y_2 - y'_2)^2 + \ldots + (y_m - y'_m)^2} \tag{17}$$

Since the values of each dimension of the model follow the same distribution, it can be set:

$$f = y_1 - y'_1 = y_2 - y'_2 = \ldots = y_m - y'_m \tag{18}$$

Substituting this into equation (17) yields

$$\Delta f = f\sqrt{k} \tag{19}$$

After the application of the Fisher sparsification mechanism, the dimensionality reduces from m to $(1-p)m$, where p represents the degree of sparsification. At this point, the real sensitivity of the parameters is given by:

$$\Delta f = f\sqrt{(1-p)k} = \Delta f\sqrt{1-p} \tag{20}$$

Therefore, Theorem 1 is proved.

Theorem 2. *In traditional federated learning, adding noise with a variance of σ^2 in each round results in a privacy budget expenditure of ϵ per round. In federated learning based on the Fisher sparsification mechanism for differential privacy, it is only necessary to add noise with a variance of $(1-p)\sigma^2$, where $p \in (0,1)$ represents the sparsity.*

Proof. Due to Theorem 1, the sensitivity reduces from Δf to $\Delta f\sqrt{1-p}$:

$$\sigma'^2 = \frac{\Delta f^2 a}{2\epsilon} = \left(\frac{\Delta f\sqrt{1-p}}{2\epsilon}\right)^2 a = (1-p)\frac{\Delta f^2 a}{2\epsilon} = (1-p)\sigma^2 \tag{21}$$

Therefore, Theorem 2 is proved.

4.3 Privacy Analysis

To achieve user-level differential privacy protection, this section analyzes the sensitivity of the aggregation process after local parameters are clipped. In Algorithm 1, the sensitivity for user-level differential privacy can be expressed as $\frac{C}{m}$. Given two adjacent rounds of parameters w_t and $w_{t,\text{adj}}$, where $w_{t,\text{adj}}$ includes an additional client:

$$\left\| \frac{1}{m}\sum_{i \in w_t} w_i^t - \frac{1}{m}\sum_{j \in w_{t,\text{adj}}} w_j^t \right\|_2 = \frac{1}{m}\|w_j^t\|_2 \leq \frac{C}{m} \tag{22}$$

In the t-th round, w_i^t and w_j^t represent the local model parameters uploaded by two different clients. Let $\tilde{w}_i^t$ and $\tilde{w}_j^t$ denote the model parameters after noise addition, where α is in the range $(1, \infty)$. The objective of this section is to compute the α-Rényi divergence between the distributions of $\tilde{w}_i^t$ and $\tilde{w}_j^t$:

$$D_\alpha(\tilde{w}_i^t \| \tilde{w}_j^t) = D_\alpha(N(\mu, \sigma^2) \| N(v, \sigma^2)) \tag{23}$$

Then, it can be calculated:

$$\begin{aligned}\exp\left((\alpha-1)D_\alpha((u,\sigma^2)\parallel N(v,\sigma^2))\right) &= \frac{1}{\sqrt{2\pi\sigma^2}}\int \exp\left(-\frac{(x-u)^2}{2\sigma^2}\right. \\ &\quad \left.-(1-\alpha)\frac{(x-u)^2}{2\sigma^2}\right)dx \\ &= \exp\left(-\frac{\alpha(\alpha-1)(u-v)^2}{2\sigma^2}\right)\end{aligned} \tag{24}$$

Hence the conclusion:

$$D_\alpha(N(u,\sigma^2)\|N(v,\sigma^2)) = \frac{\alpha(u-v)}{2\sigma^2} \le \frac{\alpha\left(\frac{C}{m}\right)^2}{2(C\sigma')^2} = \frac{\alpha}{2\sigma'^2m^2} \tag{25}$$

From the formula, it is evident that the parameter upload process after adding differential privacy noise satisfies $(\alpha, \frac{\alpha}{2\sigma'^2m^2})$-RDP. Additionally, as this algorithm employs a privacy budget allocation strategy, according to the parallel composition theorem, the differential privacy federated learning model satisfies (α,ϵ)-RDP, where $\epsilon = \max(\epsilon_1,\epsilon_2,...,\epsilon_k)$. Finally, according to the RDP to DP conversion theorem, if a random mechanism $f: D \to R$ satisfies (α,ϵ)-RDP, it follows that Algorithm 1 in this paper meets $(\epsilon + \log(1/\delta)/(\alpha-1), \delta)$-DP.

5 Evaluation

5.1 Experimental environment

The experimental setup for Section 5 is as follows, reflecting the hardware and software specifications. The detailed experimental environment is shown in Table 1.

Table 1: Experimental environment

Component	**Specification**
CPU	Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz
GPU	NVIDIA Tesla A100 SXM4 80G GPU
Memory	64GB
Storage	1200G
Operating System	Python 3.9
Framework	Pytorch 2.17

5.2 Experimental detail

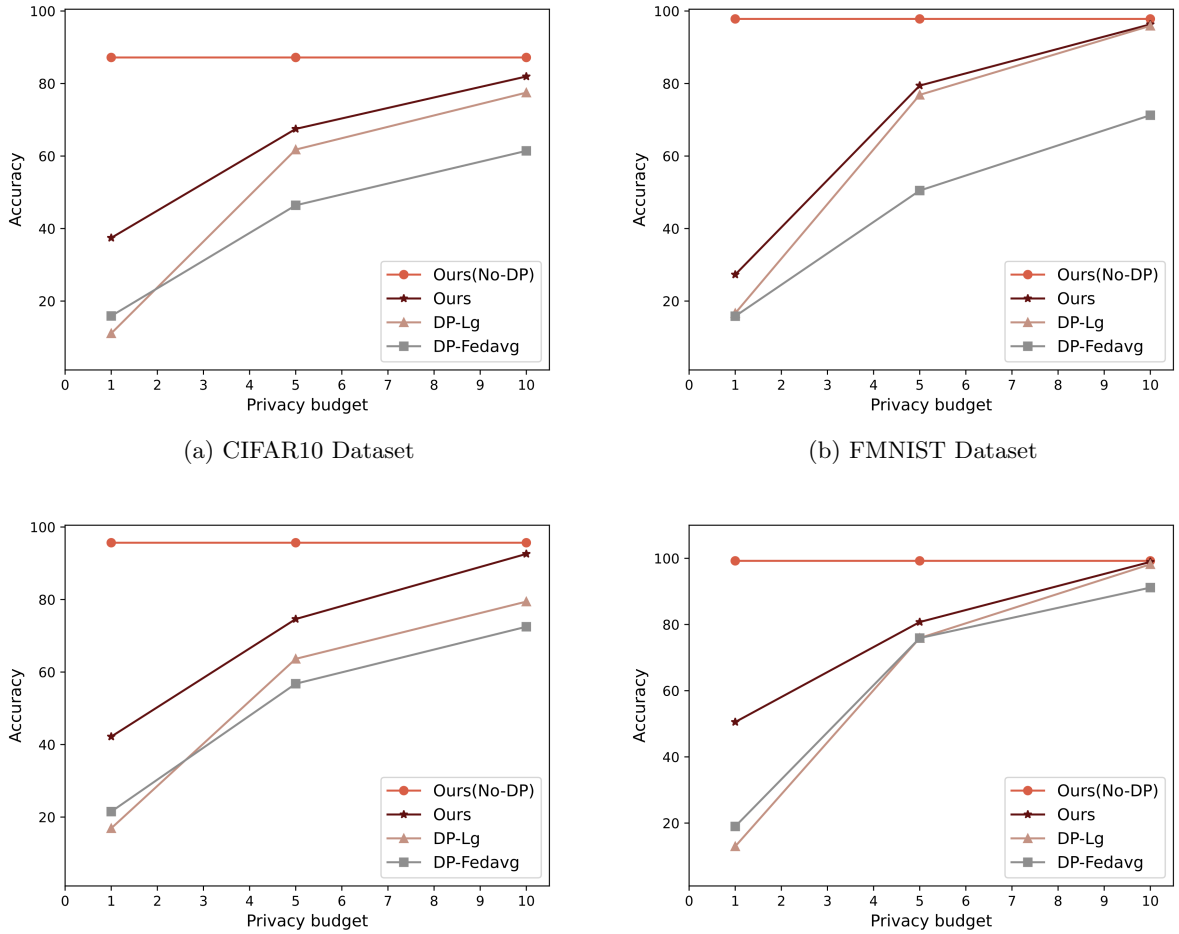
Dateset: The datasets used are CIFAR-10, FMNIST, SVHN, and MNIST.

- CIFAR-10 consists of 60,000 colour images of 32x32 pixels each, divided into 10 classes with 6,000 images per class.
- FMNIST, short for Fashion-MNIST, includes 60,000 training and 10,000 test images. These are grayscale images of 28x28 pixels, each representing one of 10 fashion categories.
- SVHN, or Street View House Numbers, comprises over 73,257 digits in real-world images, with an additional 26,032 digits for testing. The images are colour photos pre-cropped to tight bounding boxes around each digit.
- MNIST is a dataset of 60,000 training images and 10,000 test images of handwritten digits, also in grayscale and sized at 28x28 pixels.

Model: This paper uses CNN networks for training CIFAR-10, FMNIST, SVHN, and MNIST datasets. The CNN network consists of two convolutional layers and three fully connected layers. The output channels of the two convolutional layers are 6 and 16, respectively; the convolutional kernel size is 5×5, and the output channels of the three fully connected layers are 120, 84, and 10, respectively.

Heterogeneity: This paper initially randomly assigns 20% of the total available labels in the dataset to each client and then randomly distributes samples of each label to the clients that possess that label.

Hyperparameter Settings: The number of global rounds is set to 100, the number of local iterations per round is set to 5, the learning rate is set at 0.01, the number of clients is 100, the sampling rate is 0.1, and the momentum is set at 0.5.



(a) CIFAR10 Dataset

(b) FMNIST Dataset

(c) SVHN Dataset

(d) MNIST Dataset

Fig. 3: Model accuracy under different privacy budgets

5.3 Experimental result

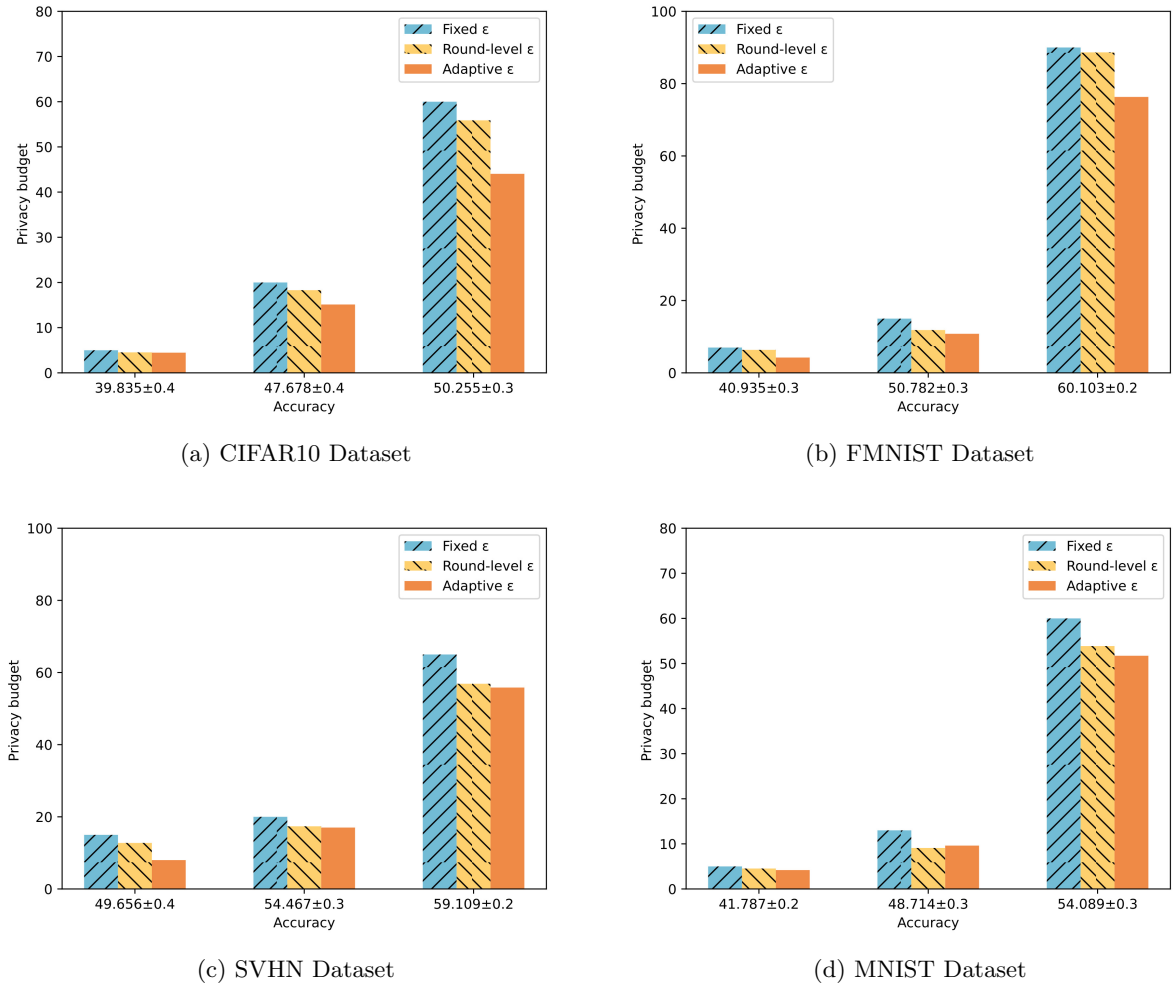
Comparison of algorithm accuracy under different privacy budgets Fig. 3 shows the average accuracy performance of four datasets, CIFAR-10, FMNIST, SVHN, and MNIST, under different privacy

budgets (ϵ=1, 5, 10) under the 20% label tilt condition. Through the analysis of these experimental graphs, a common trend can be observed in this paper: as the privacy budget ϵ increases, that is, the degree of privacy protection decreases, the model accuracy of all four data sets increases because a larger privacy budget allows the addition of less differential privacy noise to have less impact on the model accuracy. Differential privacy at the user level provides stronger data protection measures. Still, it also means that more noise must be introduced to the model parameters, so model performance is more vulnerable to the negative impact of noise. The algorithms in this paper perform better than DP-LG and DP-FedAvg algorithms in all privacy budget Settings. Specifically, in the CIFAR-10 dataset, when the privacy budget is set to 1, the algorithm in this paper shows the most significant performance improvement compared to the other two control algorithms, with a privacy budget of 5 and 10, 6% and 4% higher than the DP-LG and DP-FedAvg algorithms, respectively. It is worth noting that when the privacy budget is 10, the method's performance in this paper is close to that of the case where no privacy measures are implemented. Similar trends were observed in the FMNIST, SVHN, and MNIST datasets, except in the FMNIST dataset, where the privacy budget was set at 10, and the improvement in accuracy was less pronounced compared to the DP-LG. These experimental results demonstrate the trade-off between privacy protection and model performance; with lower privacy budgets and reduced model accuracy, stronger privacy protection can be achieved.

Table 2: Comparison of algorithm accuracy under different privacy budgets

Dataset	ϵ	**Fixed ϵ**	**Round-level ϵ**
CIFAR-10	$\epsilon = 3$	53.340	58.096
	$\epsilon = 6$	72.725	73.240
	$\epsilon = 9$	79.987	81.712
FMNIST	$\epsilon = 3$	59.171	65.531
	$\epsilon = 6$	89.035	93.534
	$\epsilon = 9$	95.656	96.227
SVHN	$\epsilon = 3$	61.400	63.463
	$\epsilon = 6$	81.369	85.931
	$\epsilon = 9$	91.291	91.324
MNIST	$\epsilon = 3$	56.142	56.968
	$\epsilon = 6$	92.220	93.443
	$\epsilon = 9$	98.104	98.576

Adaptive privacy budget allocation performance comparison Table 2 compares the average accuracy of the local model between the privacy budget allocation algorithm proposed in this paper and the average privacy budget allocation algorithm under different privacy budgets (ϵ=1, 5, 10) and the four data sets of CIFAR-10, FMNIST, SVHN, and MNIST. In the case of ϵ=3 in the CIFAR-10 dataset, the privacy budget allocation is 5% higher than the average allocated privacy budget; in the case of ϵ=3 in the FMNIST dataset, the privacy budget allocation is 6% higher than the average allocated privacy budget; in the case of ϵ=3 in the SVHN dataset, the privacy budget allocation is 2% higher than the average allocated privacy budget. The experimental results show that the performance of the privacy budget allocation algorithm between different rounds is better than the average privacy budget allocation algorithm on the four data sets, and the accuracy of the model increases with the increase of the privacy budget. This indicates that at the beginning of training, its fitting ability is weak because the model is not stable and far from the convergence point. At this stage, large noise can be accommodated without significantly impacting the direction of gradient descent, thus showing a strong anti-noise ability. With the increase in training times, the parameters gradually approach the optimal solution, and the gradient gradually decreases and eventually tends to zero. At this stage, the model becomes more stable and sensitive, the accuracy of the gradient direction is increased, and even a small amount of noise can impact the accuracy of the model. Therefore, it is necessary to increase the privacy budget in the later stage of training, and the privacy budget can be saved in the early stage to improve the model's performance in the later stage of training.



(a) CIFAR10 Dataset

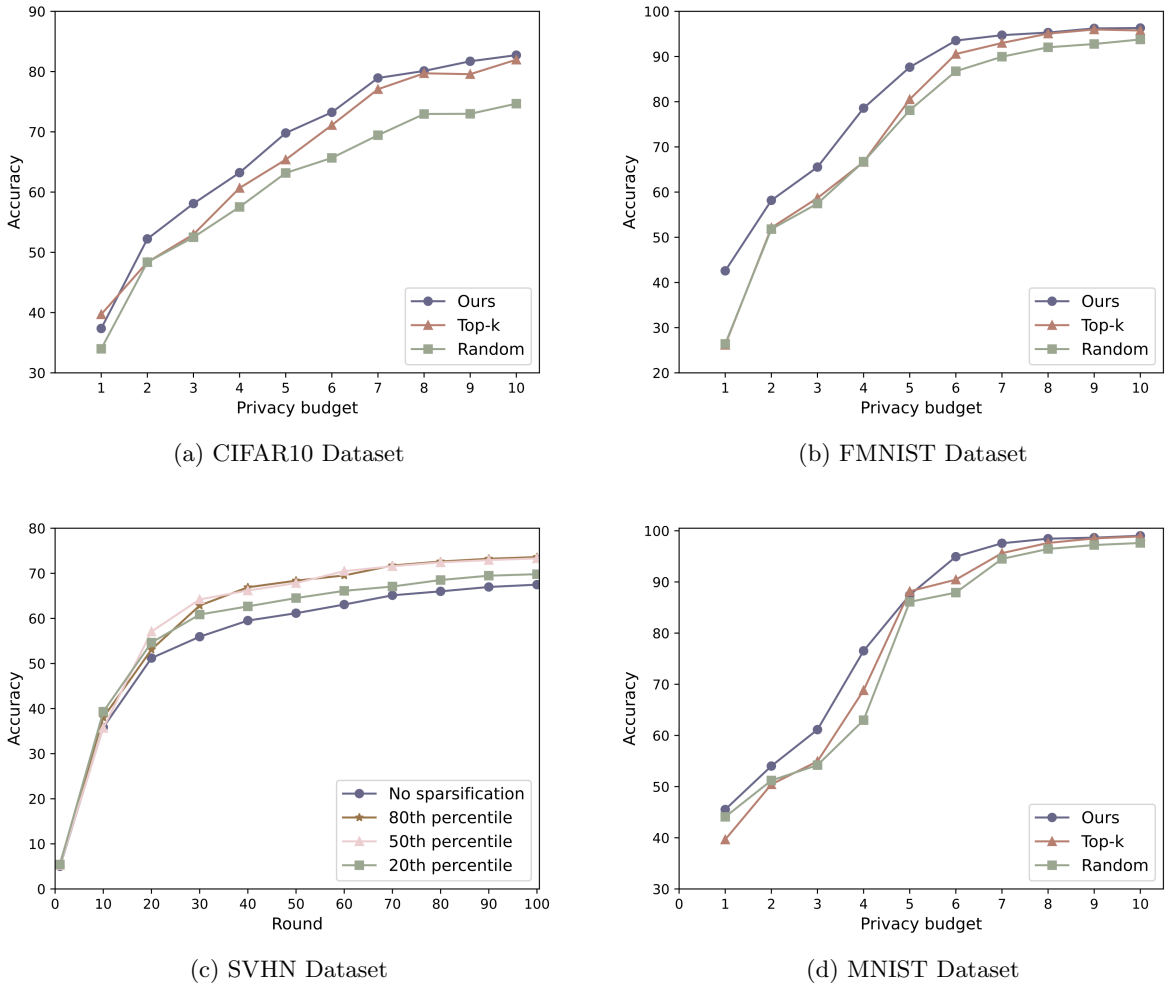
(b) FMNIST Dataset

(c) SVHN Dataset

(d) MNIST Dataset

Fig. 4: Comparison of privacy budget consumption of models with different accuracies

Fig. 4 shows that with the privacy budget set to 1, There are three algorithms: Fixed ϵ, Round-level ϵ, and Adaptive ϵ. The privacy budget is consumed by achieving the same accuracy on the CIFAR-10, FMNIST, SVHN, and MNIST four datasets. In this experiment, the privacy budget consumption of each client cannot be accumulated due to the previous client sampling, so the setting of 10 fully sampled clients is adopted to verify the privacy budget consumption effectively. By reducing the number of clients, each client has more local data, which makes each client more accurate compared to the previous setup. Experimental results show that the algorithm proposed in this paper consumes the least privacy budget under the condition of the same accuracy. For example, on the CIFAR-10 dataset, when the accuracy is about 47.678%, the adaptive privacy budget allocation algorithm consumes less than the average privacy budget allocation and the privacy budget allocation between different rounds. With the improvement of accuracy, the privacy budget is gradually reduced. When the accuracy reaches about 50.25%, less privacy budget is consumed compared to budget allocation between different rounds. Except in the MNIST data set, the privacy budget consumption during training is slightly higher than the privacy budget allocation between different rounds; the adaptive privacy budget allocation algorithm consumes the least privacy budget. The privacy budget allocation algorithm based on privacy disclosure risk can dynamically allocate a privacy budget by quantifying privacy disclosure risk. This method can dynamically optimize the balance between accuracy and privacy budget to protect privacy and realize the savings of the privacy budget.



(a) CIFAR10 Dataset

(b) FMNIST Dataset

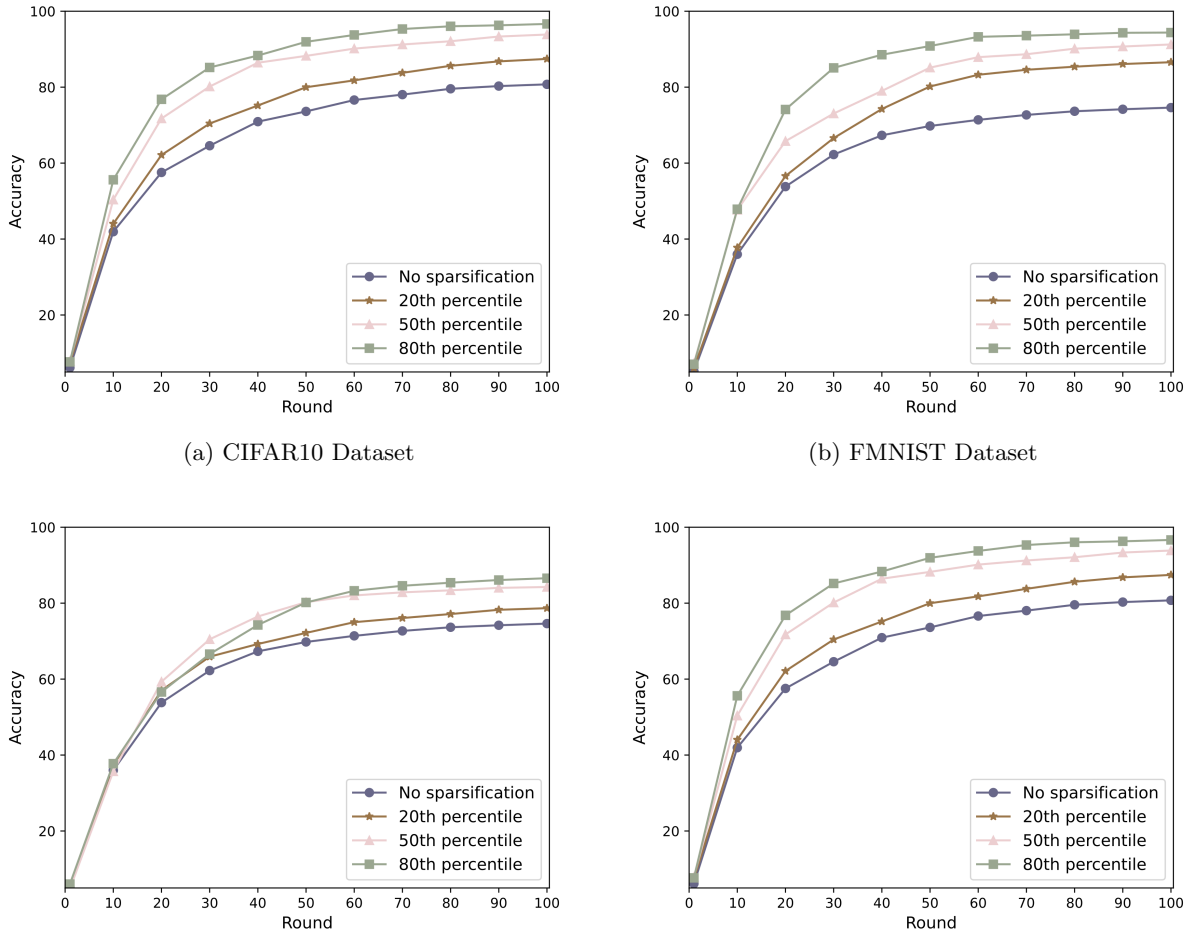
(c) SVHN Dataset

(d) MNIST Dataset

Fig. 5: Performance comparison of differential privacy model sparsification algorithms

Performance comparison of differential privacy model sparse algorithm Fig. 5 shows the accuracy performance of three different model sparse algorithms under different privacy budgets on the four data sets of CIFAR-10, FMNIST, SVHN, and MNIST. The three algorithms are model sparsity based on Fisher information (Ours), model sparsity based on maximum weight selection (Top-k), and Random model sparsity (Random). On all four datasets, the accuracy of all three algorithms improves as the privacy budget ϵ increases. On all four datasets, the sparse algorithm based on Fisher information performed lower than that of Top-k in all cases except for the CIFAR-10 data set with a privacy budget of 1. For example, in the CIFAR-10 dataset, when the privacy budget is 5, the algorithm's accuracy in this paper improves by 4% compared with the Top-k algorithm; Compared with the Random algorithm, it improves by nearly 9%. This shows that the sparse algorithm based on Fisher information is more effective in preserving important parameters in the model and reducing information loss. In contrast, the sparse strategies selected according to weights perform better than the sparse strategies of random models under higher privacy budgets. Fisher information helps preserve the parameters that have the greatest impact on performance by quantifying how important the model parameters are on the data set and how small parameter changes affect the model output. On the other hand, the sparsity of Top weights determines the importance of parameters based on the weight of parameters. It is generally believed that parameters with larger weights are more important. However, some parameters with less weight can also significantly affect model behaviour in some cases, especially when they are related to key features in the data. In contrast, random sparsity does not consider the importance of

parameters, and sparsity is performed by randomly selecting parameters, which may lead to adverse effects on the model's utility.



(a) CIFAR10 Dataset

(b) FMNIST Dataset

(c) SVHN Dataset

(d) MNIST Dataset

Fig. 6: Performance comparison of different sparsity thresholds

Fig. 6 shows the accuracy performance of different sparse thresholds on four datasets, CIFAR-10, FMNIST, SVHN, and MNIST, under the condition of a privacy budget of 5. In these four data sets, the effect of the sparse algorithm is better than that of the non-sparse algorithm, but the selection of different sparse thresholds has different effects on the accuracy of the model. This is because, on the one hand, sparsity can reduce the model norm, thus reducing the impact of sensitivity to reduce noise; on the other hand, retaining important information after sparsity can reduce the information deviation caused by clipping and obtain higher robustness. The analysis in this paper shows that selecting the 80th percentile of all Fisher information values as the sparse threshold can achieve the highest accuracy. This suggests that an optimal sparsity ratio strikes a balance between the information error introduced by sparsity and the random noise introduced by differential privacy. In the experimental environment of this paper, a larger sparse threshold has a better effect, and a higher accuracy is achieved compared with the non-sparse algorithm.

6 Conclusion

We propose the ADP-PFL algorithm to address the issue of privacy protection in personalized federated learning under data heterogeneity. The ADP-PFL algorithm introduces an adaptive privacy budget allocation mechanism that can dynamically adjust the allocation of the privacy budget based on the privacy leakage risk among clients and the number of training rounds. Moreover, it applies the Fisher information matrix to sparsify parameters, reducing errors introduced by pruning and lowering noise. Experimental results demonstrate that under the premise of ensuring privacy, the accuracy and privacy budget savings of our algorithm are superior to other algorithms.

Bibliography

[1] Jakub Konecnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 8, 2016.

[2] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022.

[3] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[4] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, 51(4):1–35, 2018.

[5] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yuan Tan. Secure multi-party computation: theory, practice and applications. *Information Sciences*, 476:357–372, 2019.

[6] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[7] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[8] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[9] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 1–11, 2019.

[10] Wenqi Wei and Ling Liu. Gradient leakage attack resilient deep learning. *IEEE Transactions on Information Forensics and Security*, 17:303–316, 2021.

[11] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 332–349. IEEE, 2019.

[12] Jiahui Hu, Zhibo Wang, Yongsheng Shen, Bohan Lin, Peng Sun, Xiaoyi Pang, Jian Liu, and Kui Ren. Shield against gradient leakage attacks: Adaptive privacy-preserving federated learning. *IEEE/ACM Transactions on Networking*, 2023.

[13] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. *Advances in Neural Information Processing Systems*, 31, 2018.

[14] Rui Hu, Yanmin Gong, and Yuanxiong Guo. Federated learning with sparsification-amplified privacy and adaptive optimization. *arXiv preprint arXiv:2008.01558*, 2020.

[15] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

[16] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. *arXiv preprint arXiv:2012.08565*, 2020.

[17] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[18] Anda Cheng, Peisong Wang, Xi Sheryl Zhang, and Jian Cheng. Differentially private federated learning with local regularization and sparsification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10122–10131, 2022.

[19] Yifan Shi, Yingqi Liu, Kang Wei, Li Shen, Xueqian Wang, and Dacheng Tao. Make landscape flatter in differentially private federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24552–24562, 2023.

[20] Xixi Huang, Ye Ding, Zoe L Jiang, Shuhan Qi, Xuan Wang, and Qing Liao. Dp-fl: a novel differentially private federated learning framework for the unbalanced data. *World Wide Web*, 23:2529–2545, 2020.

[21] Fang Dong, Xinghua Ge, Qinya Li, Jinghui Zhang, Dian Shen, Siqi Liu, Xiao Liu, Gang Li, Fan Wu, and Junzhou Luo. Padp-fedmeta: A personalized and adaptive differentially private federated meta learning mechanism for aiot. *Journal of Systems Architecture*, 134:102754, 2023.

[22] Ge Yang, Shaowei Wang, and Haijie Wang. Federated learning with personalized local differential privacy. In *2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*, pages 484–489. IEEE, 2021.

[23] Xiaoying Shen, Hang Jiang, Yange Chen, Baocang Wang, and Le Gao. Pldp-fl: Federated learning with personalized local differential privacy. *Entropy*, 25(3):485, 2023.

[24] Shuhong Chen, Jiawei Yang, Guojun Wang, Zijia Wang, Haojie Yin, and Yinglin Feng. Clfldp: Communication-efficient layer clipping federated learning with local differential privacy. *Journal of Systems Architecture*, 148:103067, 2024.

[25] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

[26] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Protect privacy from gradient leakage attack in federated learning. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 580–589. IEEE, 2022.

[27] Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul PPP Grasman, and Eric-Jan Wagenmakers. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80:40–55, 2017.

[28] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[29] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.