

Article

Not peer-reviewed version

AWDP-FL: An Adaptive Differential Privacy Federated Learning Framework

[Zhiyan Chen](#), [Hong Zheng](#)^{*}, [Gang Liu](#)

Posted Date: 20 August 2024

doi: 10.20944/preprints202408.1270.v1

Keywords: federated learning; differential privacy; adaptive weight clipping; privacy preserving



Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Article

AWDP-FL: An Adaptive Differential Privacy Federated Learning Framework

Zhiyan Chen, Hong Zheng * and Gang Liu

School of Computer Science and Engineering, Changchun University of Technology, Changchun, 130022, China; 2202203059@stu.ccut.edu.cn; Lg@ccut.edu.cn

* Correspondence: zhenghong@ccut.edu.cn

Abstract: Data security and user privacy concerns are increasingly gaining attention. Federated learning models based on differential privacy offer a distributed machine learning framework that protects data privacy; however, the added noise can impact the model's utility, making performance evaluation crucial. To optimize the balance between privacy protection and model performance, we propose the Adaptive Weight-Based Differential Privacy Federated Learning (AWDP-FL) framework. This framework processes model parameters from the perspectives of neural network layers and model weights. Initially, each participant trains the model locally. During iterative training, the clipping threshold is determined by selecting an adaptive weight coefficient, followed by adaptive gradient clipping to control the gradient magnitude. Subsequently, adaptive gradient updates are applied to the model, and dynamic Gaussian noise is introduced when uploading the parameters to protect participant privacy. The server aggregates these noise-perturbed parameters to update the global model. This framework ensures strong privacy protection while maintaining model accuracy. Theoretical analysis and experimental results validate the effectiveness of this framework under stringent privacy constraints.

Keywords: federated learning; differential privacy; adaptive weight clipping; privacy preserving

1. Introduction

Federated learning, as a distributed machine learning technique, successfully addresses the issue of data silos, enabling institutions to collaborate across regions without moving data, thereby improving the accuracy of predictive models. Researchers have greatly enhanced data privacy protection by deeply integrating various privacy-preserving technologies, including secure multi-party computation, homomorphic encryption, and differential privacy. MA et al. proposed a federated learning method based on the application of multi-key homomorphic encryption technology, aiming to achieve privacy protection while reducing computational costs [2]. PARK et al., on the other hand, utilized homomorphic encryption to directly encrypt model parameters, allowing the central server to perform computations on the ciphertext without decryption [3]. However, despite ensuring that data is "computable but not visible," these techniques significantly increase the system's computational and communication burdens [4–9]. In particular, secure multi-party computation relies on complex communication protocols, while homomorphic encryption requires a substantial number of encryption operations. In contrast, differential privacy, with its simplicity and robust privacy protection, has emerged as an important research direction in federated learning. In federated learning, differential privacy is categorized into local differential privacy and centralized differential privacy. Among these, centralized differential privacy is primarily used in client-server architectures, providing privacy protection for parameters processed and transmitted by the server. However, centralized differential privacy faces a trade-off between privacy and model performance in practical applications, and its effectiveness depends on the trustworthiness of the server. If the server improperly handles noise addition, it could lead to privacy breaches.

Local Differential Privacy and Its Applications

Local Differential Privacy (LDP) prevents privacy breaches by allowing data owners to maintain complete control over their data. Specifically, noise is added to the data locally before it is uploaded, ensuring that privacy is preserved. The LDPFL algorithm proposed by CHAMIKAPA et al. [10]

increases the randomness of input data, reducing the risk of information leakage while ensuring high-precision model training. The LDP-Fed algorithm developed by TRUEXS et al. [11] allows users to customize privacy budgets based on local privacy needs and adjust the perturbation of local model parameters. However, a major drawback of these methods is that the introduction of noise inevitably reduces model accuracy, particularly when dealing with high-dimensional data or complex models, where the impact of noise is more pronounced. To optimize differential privacy protection in federated learning, SUN et al. [12] proposed an adaptive weight parameter setting and data perturbation strategy, targeting the differences in weight layers within deep learning models to enhance model performance. The FedSGD method designed by Zhao et al. [13] perturbs gradients, not only improving model accuracy but also reducing communication overhead.

Adaptive Strategies and Their Limitations

However, the aforementioned optimization strategies may encounter applicability issues when applied to different model layers or datasets. For instance, adaptive weight strategies are highly effective when there are significant differences in inter-layer weights but may show limited performance improvement when the weights are relatively uniform. Moreover, although FedSGD reduces communication overhead, it may not effectively resist privacy attacks in low-noise environments. To balance differential privacy and model performance, LIU et al. [14] further proposed a federated learning algorithm that integrates adaptive gradient clipping, dynamically adjusting the clipping threshold to improve model accuracy. The PEDPFL algorithm by SHEN et al. [15] introduced regularization techniques to enhance model robustness. Although these methods have achieved significant results in balancing privacy protection and model performance, they are heavily dependent on parameter tuning. If the parameters are improperly set, it may lead to excessive clipping or insufficient noise injection, thereby affecting both model accuracy and privacy protection. Furthermore, the complexity of these algorithms increases the difficulty of implementation and deployment, particularly in resource-constrained environments.

Generalizability and Limitations in Specific Scenarios

To enhance communication efficiency in federated learning, LIU et al. [16] developed the APFL algorithm, which evaluates the contribution of each data attribute to the model output using a relevance propagation algorithm, and injects an appropriate amount of noise based on these contributions. WU et al. [17] adopted an adaptive learning rate adjustment strategy, enhancing the computational efficiency of local gradient descent on the client side. Wang Fangwei et al. [18] proposed an adaptive clipping differential privacy federated learning framework, optimizing model performance through dynamic clipping of gradient parameters. ZHAO et al. [19] proposed an enhanced federated learning framework that combines client-side self-sampling and adaptive data perturbation mechanisms, effectively achieving local differential privacy protection. The Fed-SMP algorithm developed by HU et al. [20] enhances privacy protection while improving communication efficiency through model sparsification techniques. The Layer-Based FL algorithm proposed by LIAN et al. [21] selectively reduces communication load by analyzing the correlation between the local model and the global model, while the robust differential privacy mechanism designed by BAEK et al. [22] effectively addresses the issue of user dropout. However, the drawback of these methods is that they generally rely on specific models or application scenarios, resulting in poor generalizability. For example, adaptive learning rates may fail when data distribution is uneven, and model sparsification techniques may not significantly reduce communication load when sparsity is low. Moreover, although the solution by BAEK et al. enhanced system robustness, it may still face the risk of exhausting the privacy budget during large-scale user dropouts.

Although these methods have improved privacy protection and performance in federated learning, they still have limitations. When dealing with model parameters, improper use of adaptive clipping may lead to excessive or insufficient clipping, thereby affecting both model performance and privacy protection. Additionally, these methods have poor generalizability and limited effectiveness when

applied to complex datasets. To address these issues, the main contributions of this paper are as follows:

- We propose an Adaptive Differential Privacy Federated Learning (AWDP-FL) framework. In each iteration of training, adaptive gradient clipping is first employed to filter parameters and determine the gradient parameters for updates. This is followed by third-moment-based adaptive gradient updates, and finally, dynamic Gaussian noise is added when uploading the parameters. The noise-perturbed model parameters are then uploaded to the server for global updates, ensuring data privacy during parameter transmission.
- We propose a weight-based adaptive gradient clipping method. First, the client computes the weight coefficients of each model parameter and uses the adaptive weight coefficients to determine the threshold for gradient clipping. Adaptive gradient clipping is then performed, followed by adaptive gradient updates to update the model parameters.
- Multiple comparative experiments were conducted on three public datasets against various state-of-the-art algorithms, and the usability of the proposed algorithm was analyzed across different metrics.

2. Relevant Technical Theories

In this chapter, we will introduce the background knowledge relevant to this study. First, Section 2.1 will provide a detailed introduction to the Federated Learning (FL) model, helping readers gain a comprehensive understanding of the entire FL process. Next, Section 2.2 will delve into the concept of differential privacy and its application in FL models, introducing some key definitions commonly used in practical applications.

2.1. Federated Learning

The development of artificial intelligence has become a major trend, the demand for high-quality data is increasing. This data often holds significant value, leading organizations and companies to be cautious about data sharing, which exacerbates the phenomenon of data silos. To address this issue, federated learning technology has emerged, allowing multiple participants to collaboratively build models without directly sharing data. Based on the distribution of participants' data, federated learning can be divided into three types: horizontal federated learning, vertical federated learning, and federated transfer learning [25]. Among these, horizontal federated learning is suitable for participants with different data samples but similar features, vertical federated learning is applicable when the participants have similar data samples but contain different features. By sharing only model parameters and other information, and collaboratively building models with the assistance of a central server, This approach reduces the risk of data leakage while maintaining model performance comparable to centralized machine learning. Federated learning primarily consists of a central server and N clients, denoted as $H = H_1, H_2, \dots, H_n$, where each client H_i holds its dataset M_i . The data samples from all clients are represented as M , satisfying the condition $\sum_{i=1}^N M_i = |M|$. The global model parameters are obtained by weighted aggregation across all clients.

$$\theta = \frac{1}{N} \sum_{i=1}^N \theta_i \quad (1)$$

The optimization problem in federated learning is as follows: $\text{Min } L(M, \theta)$, where the goal is to solve for the loss function and the minimal model parameters. In this paper, the loss function used is the cross-entropy loss function, which provides a finer-grained performance evaluation.

$$\min_{\theta} L(M, \theta) = \frac{1}{N} \sum_{i=1}^N L(M_i, \theta_i) \quad (2)$$

2.2. Differential Privacy

The fundamental principle of differential privacy [25] is to add specific noise perturbations to the original data, ensuring that changes in a single sample do not significantly affect the distribution of the output results, thereby protecting the sensitivity of individual data. By adding noise to datasets containing personal sensitive information, differential privacy effectively reduces the sensitivity of the data, transforming it into a more generalized dataset. Even if an attacker obtains certain data information, they cannot accurately identify the individual to whom the information belongs from the query results, thereby protecting personal privacy. After adopting differential privacy techniques, data remains usable while ensuring privacy protection, effectively balancing the need for data protection and application.

In the application of differential privacy in federated learning, correctly setting the gradient clipping threshold is key to balancing model performance and privacy protection. If the clipping threshold is set too high, more noise may need to be added, which could reduce the model's accuracy; if the threshold is set too low, too much information may be clipped, affecting the effectiveness of the gradients. Gradient clipping can be considered from two main aspects: first is value-based clipping, where all gradient values exceeding a fixed threshold are clipped to that threshold, precisely controlling the maximum values of all gradients in the model to ensure they do not exceed the set range. However, this method does not account for each gradient component, and when gradients are generally large, it may lead to the loss of critical gradient information. The second method is norm-based clipping, which adjusts the proportion of the entire gradient vector to meet the norm constraints, thereby preserving the direction of the original gradient vector and helping retain more information about the optimization direction. This method is currently more mainstream, but there is still room for optimization in practical applications. This paper will explore this area in depth, aiming to further optimize gradient clipping methods to better balance privacy protection and model performance.

Definition 1. Adjacent Datasets Given two datasets, M and M_1 , when the distance between M and M_1 is 1, they are referred to as adjacent datasets.

$$\|M - M_1\|_2 = 1 \quad (3)$$

The distance is defined as the sum of the absolute differences for each pair of records between the two datasets. If a corresponding record is missing in one dataset during the calculation of differences, 0 is used in place of the missing value, indicating that the two datasets are very similar.

Definition 2. (ϵ, δ) -Differential Privacy Given a random algorithm R , if for any output set S , the following inequality holds for the adjacent datasets M and M_1 , then algorithm R is said to satisfy (ϵ, δ) -differential privacy:

$$\Pr[R(M) \in S] \leq \Pr[R(M_1) \in S] \cdot e^\epsilon + \delta \quad (4)$$

Here, $\Pr[R(M) \in S]$ denotes the probability that algorithm R , when applied to dataset M , produces an output that falls within set S . The privacy budget ϵ controls the ratio of output probabilities between adjacent datasets. The smaller the ϵ , the closer the output probabilities of the two datasets, which indicates stronger privacy protection, making it more difficult for an attacker to distinguish which dataset the data came from. The parameter δ defines the probability of violating (ϵ) -differential privacy, providing flexibility in privacy protection. When $\delta = 0$, the algorithm satisfies (ϵ) -differential privacy, representing a stricter privacy guarantee standard.

Definition 3. Sensitivity The sensitivity of a function describes the maximum possible change in the function's output when a data point is added or removed from the dataset, measuring the function's responsiveness to

changes in a single data item. For two datasets, M and M_1 , differing by only one element, the sensitivity is determined by calculating the maximum possible difference in the output of function R between these two datasets, specifically:

$$\Delta s = \max_{M, M_1} \|R(M) - R(M_1)\|_2 \quad (5)$$

In the Gaussian mechanism, if the L_2 sensitivity of function R is Δs (measured using Euclidean distance), (ϵ, δ) -differential privacy is achieved by adding Gaussian-distributed random noise with mean 0 and variance σ^2 to the result.

Definition 4. Composition Theorem Suppose there is a set of algorithms $R_1(M), R_2(M), \dots, R_n(M)$, each satisfying (ϵ, δ) -differential privacy. When combined into a new algorithm $R_\vee = (R_1(M), R_2(M), \dots, R_n(M))$, the entire composite algorithm $R_\vee$ will also satisfy (ϵ, δ) -differential privacy. The privacy of individual data points in the composite algorithm remains protected, with the combination possibly being serial, parallel, etc.

Definition 5. Post-processing Immunity [25] If algorithm R satisfies (ϵ, δ) -differential privacy for the same dataset M , then even if another random algorithm A (which may not follow differential privacy principles) forms a new algorithm $B = A(R(M))$ by applying A to the output of R , the new algorithm B will still satisfy (ϵ, δ) -differential privacy.

3. Weighted Adaptive Differential Privacy Federated Learning Framework

To protect the privacy of client data in federated learning, this paper combines an adaptive weight clipping threshold selection strategy to design a novel weighted adaptive differential privacy federated learning framework (Algorithm 1). Unlike traditional federated learning frameworks, this framework introduces improvements in the way model gradients are handled, with all client operations being performed from the neural network layer perspective. During each local training iteration, clients first calculate the threshold for adaptive gradient clipping using weight coefficients, then apply adaptive gradient clipping to limit the magnitude of the gradients, followed by adaptive gradient updates to the model, and finally add dynamic Gaussian noise when uploading the parameters. This federated learning framework emphasizes an optimal balance between data privacy protection and model performance during communication, offering new perspectives and methods for privacy protection in federated learning. The communication process between the client and server is illustrated in Figure 1.

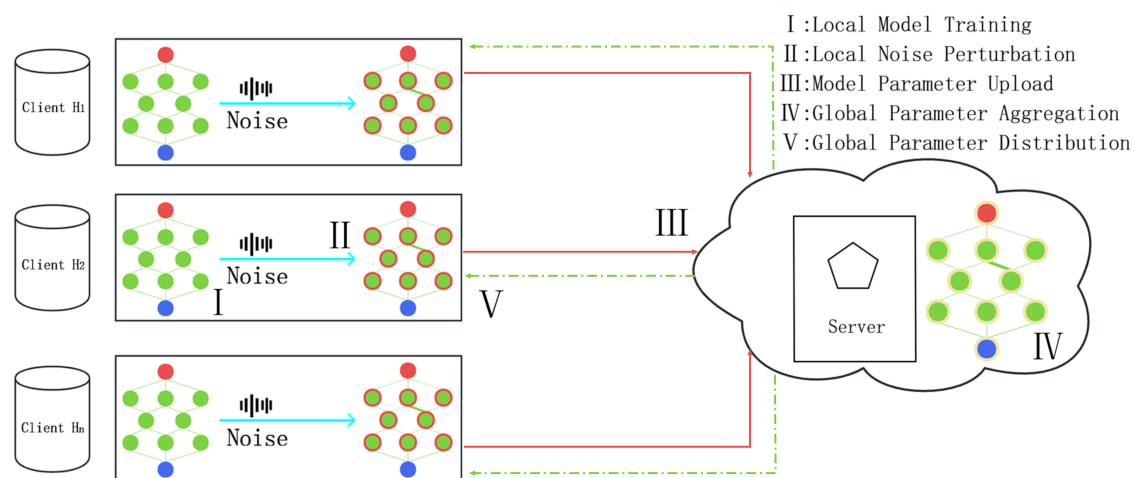


Figure 1. Federated Learning Communication Framework Diagram

3.1. Workflow of the Framework

The framework consists of four main steps:

1. **Local Model Training:** Clients must perform multiple gradient descents locally to reduce communication with the server. First, in each local iteration, the adaptive gradient clipping WDP (Algorithm 2), based on model weights, is used for gradient clipping, which enhances the model's robustness. Next, the adaptive gradient update AGU (Algorithm 3) is employed to determine the result of the model training in this iteration. Notably, this algorithm does not need to consider the training results of other clients globally; clients operate independently without affecting each other. Detailed steps will be presented in the next section.
2. **Parameter Upload:** To protect client privacy, hierarchical noise perturbation must be added to the model parameters before uploading. The model parameters processed by the WDP and AGU algorithms can reduce the noise impact caused by individual samples. This stability helps to smooth the loss function descent, accelerating convergence while minimizing the effect of noise on model parameters. The noise addition steps are as follows (using a single client as an example):

$$\begin{aligned}
 (\tilde{\theta}_t^h) &= \{(\tilde{\theta}_{t,l1}^h), (\tilde{\theta}_{t,l2}^h), \dots, (\tilde{\theta}_{t,ln}^h)\} \ni \\
 &\theta_{t,l} - \alpha \cdot \frac{\hat{V}_{t,l,d,e}^1}{\rho + \sqrt[3]{\hat{V}_{t,l,d,e}^3}} (d = |M|, e = |E|) + N(0, \sigma_l^{t^2}) \\
 &= \theta_{t,l} - \alpha \cdot \sum_{e=1}^E \frac{\hat{V}_{t,l,d,e}^1}{\rho + \sqrt[3]{\hat{V}_{t,l,d,e}^3}} (d \in M) \\
 &= \theta_{t,l} - \alpha \cdot \sum_{e=1}^E \sum_{d=1}^{\lceil M/B \rceil} \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{t,l,d,e}^3}} (d = B) \\
 &= \theta_{t,l} - \frac{\alpha}{|B|} \cdot \sum_{e=1}^E \sum_{d=1}^{\lceil M/B \rceil} \sum_{b=1}^B \frac{\hat{V}_{t,l,d,e}^1}{\rho + \sqrt[3]{\hat{V}_{t,l,d,e}^3}}
 \end{aligned} \tag{6}$$

Here, L represents the neural network layer level, which serves as the basis for all operations in this paper. $(\tilde{\theta}_t^h)$ denotes the noise-perturbed model parameters, $\theta_{(t,l)}$ are the unperturbed model parameters, M represents the client data, B is the training data for each batch, t is the current global round, d is the current training data, and e is the current batch's local iteration. $\hat{V}_{(t,l,d,e)}^1 / (\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3})$ represents the model gradient of the l -th layer in the t -th round after adaptive momentum update. $N(0, \sigma^2)_l^t$ is the noise perturbation added before uploading, which satisfies $N(0, (\Delta s \sigma)^2)_l^t$, where Δs_l^t is the sensitivity determined by the adaptive clipping threshold.

3. **Global Parameter Aggregation:** After the local training of sampled clients is completed, the server performs weighted aggregation of the noise-perturbed parameters uploaded by the clients to update the global model for the next round, specifically as follows:

$$\tilde{\theta}_t = \sum_{h=1}^H \tilde{\theta}_t^h \tag{7}$$

4. **Global Parameter Distribution:** In the federated learning framework, the server employs a selective parameter broadcasting strategy. Specifically, the server does not broadcast the latest model parameters to all clients every time; instead, it randomly selects a portion of clients for parameter updates. This approach avoids redundant broadcasting and reduces unnecessary communication overhead. During this process, the server does not need direct access to the clients' local data. After receiving the global model from the server, clients adaptively update their local models based on these parameters, maintaining the efficiency of model updates.

Client noise perturbation depends on sensitivity, sensitivity depends on the determination of the clipping threshold, and the clipping threshold depends on adaptive weight selection. Gradient clipping ensures that the model norm remains within a certain range. Specifically:

$$L_{t,l} = \|g_t(x_t)\|_2 \leq \text{Clip}_{t,l} \quad (8)$$

Thus, the sensitivity calculation is as shown in Lemma 1:

Lemma 1. $\tilde{\theta}_t^h(d = |M|)$ represents the model parameters of client h after local training on a dataset M of size $|M|$. Therefore, the sensitivity is:

$$\Delta s_l^t = \left\| \tilde{\theta}_t^h(d = |M|) - \tilde{\theta}_t^h(d = |M'|) \right\| \leq 2\alpha \text{Clip}_{t,l} \frac{E}{B} \quad (9)$$

Proof. $\tilde{\theta}_t^h(d = |M|)$ represents the model parameters obtained after E local iterations from $\theta_t^h(d = |M|)$. $|M|$ and $|M'|$ are adjacent datasets of client h , with their only difference being in batch B from $|M|$ and batch B' from $|M'|$. Therefore, the standard deviation of the hierarchical Gaussian noise to be added before uploading the model parameters can be calculated as follows:

$$\begin{aligned} \Delta s_l^t &= \left\| \tilde{\theta}_t^h(d = |M|) - \tilde{\theta}_t^h(d = |M'|) \right\|_2 \\ &= \left\| \theta_{t,l} - \frac{\alpha}{|B|} \cdot \sum_{e=1}^E \sum_{d=1}^{\lceil M/B \rceil} \sum_{b=1}^B \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}} - \theta_{t,l} - \frac{\alpha}{|B|} \cdot \sum_{e=1}^E \sum_{d=1}^{\lceil M'/B \rceil} \sum_{b=1}^B \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}} \right\|_2 \\ &= \frac{\alpha}{|B|} \left\| \sum_{e=1}^E \left(\sum_{d=1}^{\lceil M/B \rceil} \sum_{b=1}^B \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}} - \sum_{d=1}^{\lceil M'/B \rceil} \sum_{b=1}^B \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}} \right) \right\|_2 \\ &= \frac{\alpha}{|B|} \sum_{e=1}^E \left\| \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}}(d = |M|) - \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}}(d = |M'|) \right\|_2 \\ &\leq \frac{\alpha}{|B|} \left(\sum_{e=1}^E \left\| \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}}(d = |M|) \right\|_2 + \sum_{e=1}^E \left\| \frac{\hat{V}_{(t,l,d,e)}^1}{\rho + \sqrt[3]{\hat{V}_{(t,l,d,e)}^3}}(d = |M'|) \right\|_2 \right) \\ &\leq \frac{2\alpha \text{Clip}_{t,l} E}{B} \end{aligned} \quad (10)$$

□

3.2. Weight-Based Adaptive Gradient Clipping and Update

In traditional methods, the gradient clipping threshold is usually determined based on researchers' experience and pretraining on public datasets. This fixed threshold approach aims to ensure model convergence and performance but fails to account for differences between various layers of the neural network, which can negatively impact model performance in certain cases. With technological advancements, a new approach has been developed that dynamically adjusts the clipping threshold using the L_2 norm percentile of historical gradients, allowing for more flexible responses to gradient changes. However, despite this method's increased flexibility in threshold adjustment, it may not effectively address situations where model training performs poorly, leading to wasted computational resources during the update process and a lack of personalized settings for different network layers. To address this issue, this paper proposes a weight-based adaptive gradient clipping and update algorithm, with the detailed implementation steps as follows:

A. Layer-wise Adaptive Weight Coefficients

Algorithm 1 Weighted Adaptive Differential Privacy Federated Learning Framework

Input: Initial model parameters θ_t , learning rate α , client set O , randomly selected clients for training H , communication rounds between client and server T , number of neural network layers L , batch size B , client data M , adaptive clipping history norm sequence P , number of local iterations E , momentum update parameters τ_1^t, τ_2^t, ρ .

Output: Model parameters $\tilde{\theta}_t$

```

1: for  $t = 1$  to  $T$  do
2:   # Client Side
3:   for  $h = 1$  to  $H$  do
4:     Init  $\theta_t \leftarrow \theta_t$  // Broadcast global parameters to update the local model
5:     for  $e = 1$  to  $E$  do
6:       for  $B_{i,h} \in M_h$  do //  $i$  is an integer from 1 to  $\lceil M/B \rceil$ 
7:          $G_{t,l} \leftarrow g_t(x_t)$ 
8:          $G_{t,l} \leftarrow \text{WDP}(G_{t,l})$ 
9:          $\theta_t \leftarrow \text{AGU}(G_{t,l})$ 
10:      end for
11:    end for
12:     $\theta_t^h \leftarrow \theta_t + N(0, \sigma^2)$  // Noise perturbation
13:  end for
14:  # Server Side
15:   $\bar{\theta}_t \leftarrow \frac{1}{|H|} \sum_{h=1}^H \theta_t^h$  // Server-side parameter aggregation
16: end for

```

Calculate the gradient and L_2 norm for each neural network layer during each iteration, where $l \in \{1, 2, 3, \dots, L\}$, and L is the number of neural network layers.

$$G_{t,l} = g_t(x_t) \quad (11)$$

$$L_{t,l} = \|g_t(x_t)\|_2 \quad (12)$$

Compute the mean of all gradients for each layer during each iteration as the preprocessing weight parameter τ_1 .

$$\overline{G}_{t,l} = \frac{1}{|B|} \sum_{t=1}^B G_{t,l} \quad (13)$$

Add this value layer-wise to a sequence, generating a new weight preprocessing historical gradient sequence GB .

$$GB = \{\overline{G}_{B_1,l}, \overline{G}_{B_2,l}, \overline{G}_{B_3,l}, \dots, \overline{G}_{B_m,l}\} \quad (14)$$

Calculate the mean of GB layer-wise as the preprocessing weight parameter τ_2 .

$$\overline{GB}_{t,l} = \frac{1}{|m|} \sum_{m=1}^n \overline{G}_{B_m,l} \quad (m \geq 2) \quad (15)$$

Use the local weight parameter τ_1 and the global weight parameter τ_2 to calculate the weight coefficient $GW_{B,l}$ layer-wise according to the formula.

$$GW_{t,l} = \cos(\overline{G}_{t,l}, \overline{GB}_{t,l}) = \frac{\overline{G}_{t,l} \cdot \overline{GB}_{t,l}}{\|\overline{G}_{t,l}\| \|\overline{GB}_{t,l}\|} \quad (16)$$

B. Adaptive Clipping Threshold Selection

In this framework, a high cosine similarity between two updates indicates that the direction of gradient updates remains consistent, suggesting a stable training process. Conversely, low or negative similarity may indicate issues such as gradient explosion or vanishing gradients. Here, B represents the batch size, and M is the number of training data points in the client.

Weight the norm of each iteration according to the layer-wise weight coefficient $GW_{B,l}$.

$$\text{real}L_{t,l} = \left(\max_{0 \leq t \leq B, 0 \leq l \leq L} L_{t,l} - \frac{1}{|B|} \sum_{t=1}^B L_{t,l} \right) / 2 \cdot \text{GW}_{t,l} + \frac{1}{|B|} \sum_{t=1}^B L_{t,l} \quad (17)$$

Add the weighted gradient norms of this batch to a sequence layer-wise, resulting in the threshold processing historical norm list LL .

$$LL = \{\text{real}L_{1,l}, \text{real}L_{2,l}, \text{real}L_{3,l}, \dots, \text{real}L_{M,l}\} \quad (18)$$

Calculate the percentile P of the LL sequence layer-wise as the clipping threshold for this iteration.

$$\text{Clip}_{t,l} = \{LL\}_P = \{\text{real}L_{1,l}, \text{real}L_{2,l}, \text{real}L_{3,l}, \dots, \text{real}L_{M,l}\}_P \quad (19)$$

Perform layer-wise gradient clipping on each data point in this iteration.

$$\text{real}G_{t,l} = \frac{G_{t,l}}{\max(1, L_{t,l} / \text{Clip}_{t,l})} \quad (20)$$

Take the mean layer-wise as the gradient for this batch iteration, preparing for the gradient update.

$$\tilde{G}_{t,l} = \frac{1}{|B|} \sum_{t=1}^B \text{real}G_{t,l} \quad (21)$$

Algorithm 2 Weight-Based Adaptive Gradient Clipping (WDP)

Input: Unupdated model parameters after training θ_t , number of neural network layers L , communication rounds between client and server T , batch size B , client data M , adaptive clipping history norm sequence P .

Output: Processed model parameters $\tilde{\theta}_t$.

- 1: $\bar{G}_{t,l} = \frac{1}{|B|} \sum_{t=1}^B G_{t,l}$
 - 2: $\bar{G}_{t,l} = \frac{1}{|B|} \sum_{m=1}^n G_{B_m,l}$ ($m \geq 2$)
 - 3: # Compute layer-specific weight coefficients
 - 4: $\text{GW}_{t,l} = \cos(\bar{G}_{t,l}, \bar{G}_{t,l}) = \frac{\bar{G}_{t,l} \cdot \bar{G}_{t,l}}{\|\bar{G}_{t,l}\| \cdot \|\bar{G}_{t,l}\|}$
 - 5: # Compute weighted norm
 - 6: $\text{real}L_{t,l} = \frac{(\max_{0 \leq t \leq B, 0 \leq l \leq L} L_{t,l} - \frac{1}{|B|} \sum_{t=1}^B L_{t,l})}{2 \cdot \text{GW}_{t,l}} + \frac{1}{|B|} \sum_{t=1}^B L_{t,l}$
 - 7: # Compute clipping threshold:
 - 8: $\text{Clip}_{t,l} = \{LL\}_P = \{\text{real}L_{1,l}, \text{real}L_{2,l}, \text{real}L_{3,l}, \dots, \text{real}L_{(M,l)}\}_P$
 - 9: # Layer-specific gradient clipping
 - 10: $\tilde{G}_{t,l} = \frac{1}{|B|} \sum_{t=1}^B G_{t,l} / \max(1, \frac{L_{t,l}}{\text{Clip}_{t,l}})$
-

C. Adaptive Momentum Gradient Updates

Initialize the first-order moment V_t^1 and third-order moment V_t^3 based on historical gradients, and define momentum decay parameters τ_1^t , τ_3^t , and hyperparameter ρ . The third-order moment measures the symmetry of the data distribution; if the data distribution is symmetric, the skewness is zero. If the distribution is skewed to the left or right, the skewness is non-zero. Using the first-order and third-order moments based on historical gradients allows the parameter update to accumulate past gradient directions, helping to escape local minima or saddle points. Momentum aids the algorithm in accelerating towards the optimal solution and reduces oscillations in directions with smaller gradients, thereby speeding up convergence.

Calculate the first-order moment estimate:

$$V_t^1 = \tau_1^t \cdot V_{t-1}^1 + (1 - \tau_1^t) \cdot \tilde{G}_{t,l} \quad (22)$$

Calculate the third-order moment estimate:

$$V_t^3 = \tau_3^t \cdot V_{t-1}^3 + (1 - \tau_3^t) \cdot (\tilde{G}_{t,l})^3 \quad (23)$$

Based on the initial gradient estimates, which often have large variance and bias, the correction mechanism adjusts the momentum estimates to more accurately reflect the true gradient information.

Correct the first-order moment:

$$\hat{V}_t^1 = \frac{V_t^1}{1 - \tau_1^t} \quad (24)$$

Correct the third-order moment:

$$\hat{V}_t^3 = \frac{V_t^3}{1 - \tau_3^t} \quad (25)$$

Update the model parameters:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{V}_t^1}{\rho + \sqrt[3]{\hat{V}_t^3}} \quad (26)$$

The processing of historical gradients using the third-order moment provides higher-dimensional data analysis capabilities, dynamically refining gradient information. This better utilizes model gradients processed by weighted adaptive gradient clipping, and the adaptive momentum update produces more complex model parameters, enhancing the model's performance.

Algorithm 3 Adaptive Gradient Update (AGU)

Input: Adaptively clipped model parameters $\tilde{G}_{t,l}$, learning rate α , communication rounds between client and server T , number of neural network layers L , momentum update parameters τ_1^t, τ_2^t, ρ .

Output: Model parameters $\hat{\theta}_t$.

```

1: # Gradient Update
2:  $\hat{V}_t^1 \leftarrow \tau_1^t \cdot \hat{V}_{t-1}^1 + (1 - \tau_1^t) \cdot \tilde{G}_{t,l}$  // Moment Estimation
3:  $\hat{V}_t^3 \leftarrow \tau_3^t \cdot \hat{V}_{t-1}^3 + (1 - \tau_3^t) \cdot \tilde{G}_{t,l}^3$ 
4:  $\hat{V}_t^1 \leftarrow \frac{\hat{V}_t^1}{1 - \tau_1^t}$  // Corrected Moment Estimation
5:  $\hat{V}_t^3 \leftarrow \frac{\hat{V}_t^3}{1 - \tau_3^t}$ 
6:  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{V}_t^1}{\rho + \sqrt[3]{\hat{V}_t^3}}$  // Adaptive Gradient Update

```

3.3. Privacy Analysis

Client Parameter Upload Phase: In the t -th round of local training, each participating client h adds Gaussian noise (with a mean of 0 and a standard deviation of σ) to the model parameters θ_t^h obtained by minimizing the loss function, where $\theta_t^h = \arg \min L(M_h, \theta_t)$, and θ_t is the global model parameter in the t -th round. The standard deviation σ is calculated according to Equation (10), $\Delta s_l^t \leq 2\alpha \text{Clip}_{t,l} \frac{\epsilon}{B}$, where Δs_l^t represents the sensitivity of client i in the l -th layer during the t -th round. In terms of privacy analysis, the client uploads the noise-added model parameters ($\tilde{\omega}_t^i$) to the server, and the client satisfies (ϵ, δ) -differential privacy.

Server Parameter Distribution Phase: In each round, the server performs weighted aggregation of the noise-added model parameters from the participating clients and updates the global model. According to Differential Privacy Definition 5 (Post-processing Immunity [25]), the server's parameter distribution process in each round satisfies (ϵ, δ) -differential privacy. After T rounds of communication between the client and server, the client's dataset has been accessed T times in sequence, forming a parallel sequence combination. According to Differential Privacy Definition 4 (Composition Theorem), the privacy budget equals the maximum of all client privacy budgets, i.e., $T\epsilon$. The algorithm satisfies $(T\epsilon, T\delta)$ -differential privacy.

4. Experiments

4.1. Experimental Setup

Datasets: Three public datasets are used

- **MNIST Dataset:** Based on a single-channel handwritten digit recognition dataset, it contains images of digits from 0 to 9, with each image being a 28×28 pixel grayscale image. The dataset includes 60,000 images for training and 10,000 images for testing.
- **Fashion-MNIST:** Based on a single-channel fashion classification dataset, it contains 70,000 grayscale images of 10 different types of clothing, with each image being 28×28 pixels. The classification for testing and validation is the same as the MNIST dataset but is more challenging.
- **CIFAR-10 Dataset:** Based on three-channel color images of 10 classes, such as ships, airplanes, cars, etc., each image is 32×32 pixels. The dataset includes 50,000 images for training and 10,000 images for testing.

Network Structure

The network model for the MNIST and Fashion-MNIST datasets includes one convolutional layer and one fully connected layer, with 256 output neurons. Additionally, a ReLU activation layer is applied using a built-in activation function to enhance the model's nonlinear expression capability. For the CIFAR-10 dataset, the network structure includes two convolutional layers, each followed by a ReLU activation layer and a max-pooling layer to enhance feature extraction and reduce data dimensionality. The features are then converted through two fully connected layers, from 384 input features to 192 output features, to further extract and combine useful information.

Experimental Setup

The experiments were conducted using the Pytorch framework on an NVIDIA GeForce RTX 4090 server, with a CPU configuration of 16 vCPUs (Intel(R) Xeon(R) Gold 6430). The results were averaged over multiple tests.

4.2. Experimental Results and Analysis

This section aims to validate the effectiveness of the algorithm. We will compare the AWDP-FL algorithm with the baseline federated learning algorithm without differential privacy (NoDP-FL)[1] and three advanced federated learning differential privacy algorithms: (1) Client-level Differential Privacy Federated Learning (CDP-FL) [16]; (2) Federated Learning Differential Privacy with Adaptive Learning Rate (FDP-FL) [17]; (3) Adaptive Differential Privacy Federated Learning (ADP-FL) [18]. In the experiments, we selected 50 clients, with accuracy expressed as a percentage. The parameters used by each algorithm were kept consistent, such as the number of local iterations, global communication rounds, and batch sampling size. Since the results of a single metric are insufficient to fully reflect the robustness of an algorithm, in practical applications, it is possible for two algorithms to achieve the same model accuracy while consuming different amounts of privacy budget. Therefore, our experimental content is divided into two parts: comparison of model accuracy and comparison of model loss function values. The following subsections will independently analyze these two metrics.

4.2.1. Model Accuracy Analysis

Model accuracy reflects the model's classification ability on the test dataset, i.e., the proportion of samples correctly classified by the model. High accuracy indicates good classification performance on the test set, providing a more intuitive reflection of the overall model performance. In selecting the privacy budget, we did not fixate on a single value but chose an arithmetic sequence. For example, in the MNIST dataset, the sequence used was 0.4, 0.6, and 0.8. The reason for this is that different privacy budgets can more clearly demonstrate the robustness of different algorithms. If the model accuracy shows little variation across this sequence, it indicates that the algorithm can adapt to smaller privacy budgets; if the model accuracy fluctuates significantly, it suggests that the algorithm has reached its

maximum adaptable privacy budget within this range. Using a range of privacy budgets also allows for a clearer demonstration of each algorithm’s performance under different conditions, establishing the critical point of the privacy budget for that dataset. Tables 1, 2, and 3 respectively show the model accuracy of different algorithms across the three datasets. Next, we will analyze the experimental results for each dataset individually.

Table 1. Model Accuracy of Different Algorithms on the MNIST Dataset

Algorithm / Privacy Budget	$\epsilon = 0.4$	$\epsilon = 0.6$	$\epsilon = 0.8$
NoDP-FL [1]		89.84	
CDP-FL [16]	78.01	79.32	79.85
ADP-FL [18]	83.64	83.80	84.01
FDP-FL [17]	79.25	82.48	84.17
AWDP-FL	87.59	87.96	88.10

Table 2. Model Accuracy of Different Algorithms on the Fashion-MNIST Dataset

Algorithm / Privacy Budget	$\epsilon = 0.4$	$\epsilon = 0.6$	$\epsilon = 0.8$
NoDP-FL [1]		84.53	
CDP-FL [16]	77.01	77.12	77.58
ADP-FL [18]	79.02	79.40	79.45
FDP-FL [17]	79.12	79.46	79.52
AWDP-FL	82.13	82.76	83.22

Table 3. Model Loss Function Values of Different Algorithms on the CIFAR-10 Dataset

Algorithm / Privacy Budget	$\epsilon = 4$	$\epsilon = 5$	$\epsilon = 6$
NoDP-FL [1]		76.99	
CDP-FL [16]	53.05	53.37	53.65
ADP-FL [18]	66.72	67.45	70.00
FDP-FL [17]	59.44	61.56	61.87
AWDP-FL	70.00	72.00	74.42

In the MNIST dataset, the number of communication rounds is set to 50, as fewer communication rounds can reduce the consumption of the privacy budget. The privacy budget is selected within the range of 0.4 to 0.8, with smaller privacy budgets providing a higher level of privacy protection. As shown in Figure 2a-2c, the NoDP-FL algorithm converges within 10 to 20 rounds, due to the lack of noise perturbation, which provides limited privacy protection. It serves as the experimental baseline to validate the effectiveness of other algorithms. Under privacy budgets of 0.4 and 0.8, the model accuracies for the NoDP-FL and AWDP-FL algorithms are 89.84%, 89.84% and 87.59%, 88.10%, respectively. This indicates that as the privacy budget decreases and the perturbation increases, the model accuracy also decreases accordingly, but the error remains within 1.74%. Under the same conditions, the highest model accuracy among the other algorithms is achieved by the FDP-FL algorithm, reaching 84.17%. This demonstrates that the AWDP-FL algorithm improves the maximum model accuracy by approximately 4% compared to other algorithms, proving it achieves a better balance between privacy protection and model performance.

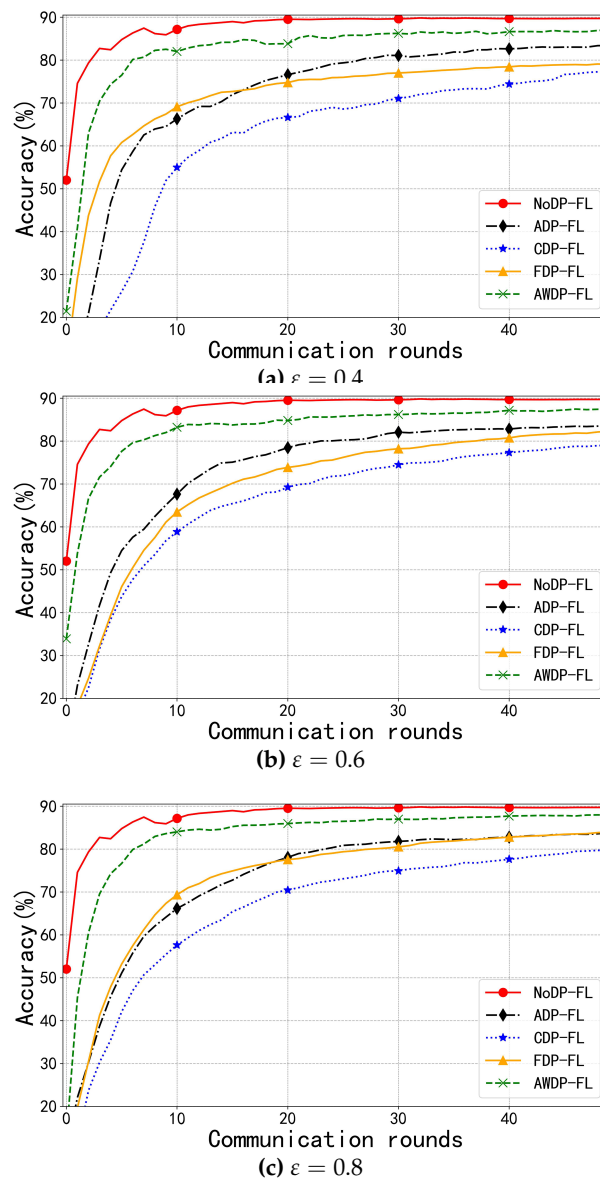


Figure 2. Model Accuracy on the MNIST Dataset at Different Privacy

In the Fashion-MNIST dataset, the number of communication rounds is set to 100, and the privacy budgets are selected as 0.4, 0.6, and 0.8. A greater number of communication rounds helps to more clearly determine the convergence interval. As shown in Figure 3a-3c, the model accuracy of the NoDP-FL algorithm is 84.53%, while the model accuracy of the AWDP-FL algorithm gradually approaches that of the NoDP-FL algorithm as the privacy budget increases. At a privacy budget of 0.8, the accuracy of AWDP-FL is 83.22%, only 1.31% lower than the NoDP-FL algorithm. The model accuracies of other algorithms, FDP-FL [17], ADP-FL [18], and CDP-FL [16], are 79.52%, 79.45%, and 77.58%, respectively. It can be seen that the model accuracy of AWDP-FL is higher than that of the other algorithms, with a maximum improvement of 5.64%. This indicates that under higher privacy budgets, the algorithm shows a stable improvement in accuracy.

Regarding the convergence interval, under the three privacy budgets, the NoDP-FL algorithm converges within 50 rounds, while the AWDP-FL algorithm converges within 70 to 80 rounds, as shown in Figure 3d. The other algorithms, however, require more than 90 rounds to converge. This indicates that our algorithm can reduce communication time compared to other algorithms, as well as reduce privacy budget consumption at the communication level.

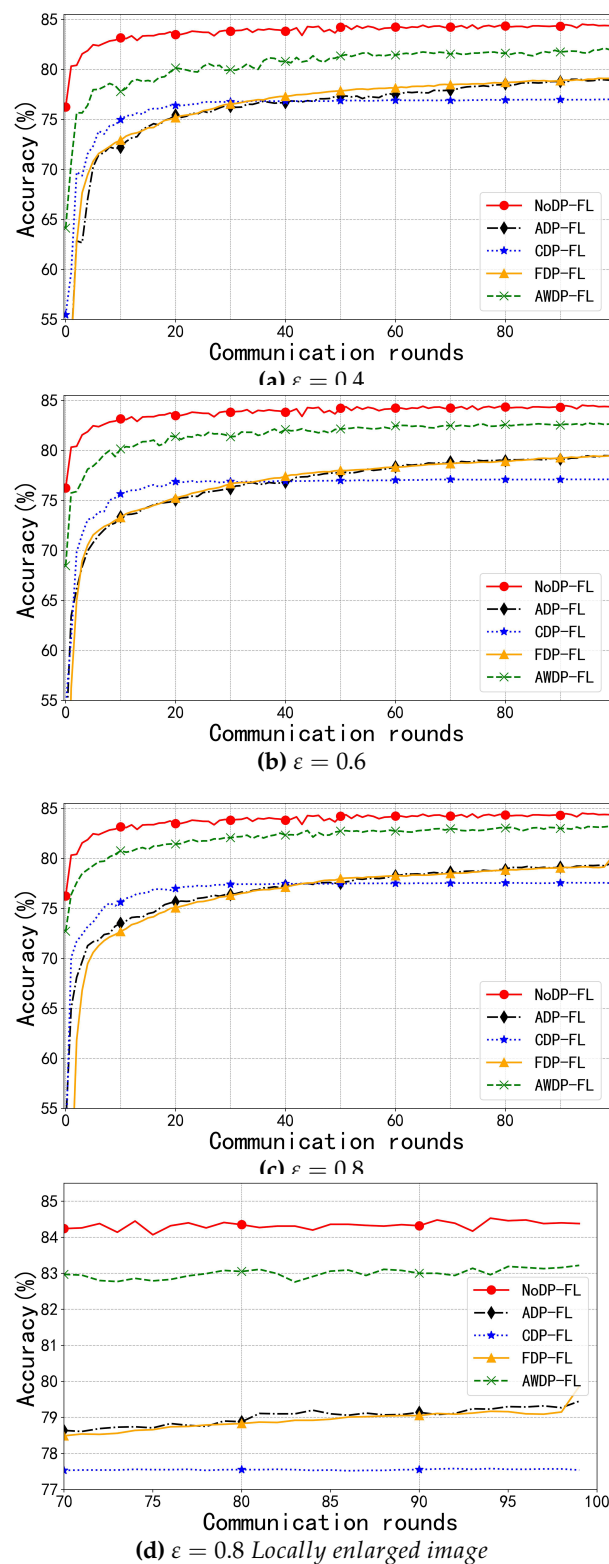


Figure 3. Model Accuracy on the Fashion-MNIST Dataset at Different Privacy

In the CIFAR-10 dataset, the number of communication rounds is also set to 100, and the privacy budgets are chosen as 4, 5, and 6. This is because CIFAR-10 is a more complex dataset, containing three-channel color images. As shown in Figure 4a-4c, with a privacy budget of 6, the AWDP-FL algorithm exhibits higher accuracy than the NoDP-FL algorithm during rounds 5 to 20 of communication. This is because, in the early stages of training, AWDP-FL's rapid convergence outperforms the NoDP-

FL algorithm, resulting in faster convergence. However, as the number of communication rounds increases, the accuracy gap between the two algorithms gradually narrows, with a final difference of 2.57%. Compared to other algorithms, with the highest accuracy being 61.87%, AWDP-FL significantly improves model performance, further demonstrating the superiority of the AWDP-FL algorithm.

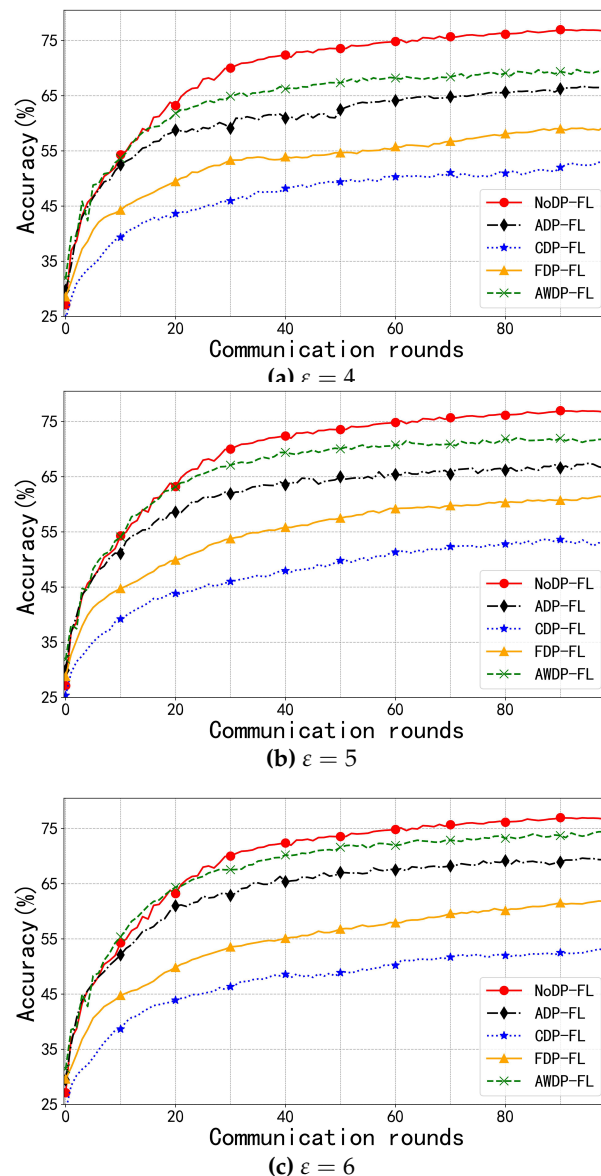


Figure 4. Model Accuracy on the MNIST Dataset at Different Privacy

4.2.2. Model Loss Analysis

In many practical applications, accuracy is often the preferred performance metric. However, in datasets with class imbalances, accuracy can lead to misleading conclusions. For example, in a dataset where 90% of the samples belong to class A and 10% belong to class B, a model that always predicts class B, a model that always predicts class A would still achieve 90% accuracy, but this is clearly not a good model. Accuracy only considers whether the prediction is correct, without taking into account the confidence of the prediction. The loss function is used to measure the difference between the model's predictions and the true values; the smaller the loss value, the closer the model's predictions are to the true values. This paper uses the cross-entropy loss function, which provides a finer-grained performance evaluation by not only considering whether the prediction is correct but also taking into account the degree of bias in the prediction. Based on this, we compared the loss function values of various algorithms under different

privacy budgets, using the same parameters and configurations as in the previous section. Tables 4, 5, and 6 respectively show the model loss function values of different algorithms across the three datasets. Next, we will analyze the experimental results for each dataset individually.

Table 4. Model Loss Function Values of Different Algorithms on the MNIST Dataset

Algorithm / Privacy Budget	$\epsilon = 0.4$	$\epsilon = 0.6$	$\epsilon = 0.8$
NoDP-FL [1]		0.3601	
CDP-FL [16]	0.8341	0.8013	0.7994
ADP-FL [18]	0.6734	0.6709	0.6686
FDP-FL [17]	1.0365	0.6180	0.5644
AWDP-FL	0.4384	0.4389	0.4323

Table 5. Model Loss Function Values of Different Algorithms on the Fashion-MNIST Dataset

Algorithm / Privacy Budget	$\epsilon = 0.4$	$\epsilon = 0.6$	$\epsilon = 0.8$
NoDP-FL [1]		0.4395	
CDP-FL [16]	0.6817	0.6634	0.6540
ADP-FL [18]	0.6125	0.5977	0.5976
FDP-FL [17]	0.6097	0.6062	0.5959
AWDP-FL	0.5701	0.5197	0.5013

Table 6. Model Loss Function Values of Different Algorithms on the CIFAR-10 Dataset

Algorithm / Privacy Budget	$\epsilon = 4$	$\epsilon = 5$	$\epsilon = 6$
NoDP-FL [1]		0.6754	
CDP-FL [15]	1.4076	1.3810	1.3744
ADP-FL [17]	1.0303	1.0383	0.9643
FDP-FL [16]	1.2275	1.1925	1.1844
AWDP-FL	0.9204	0.8488	0.7916

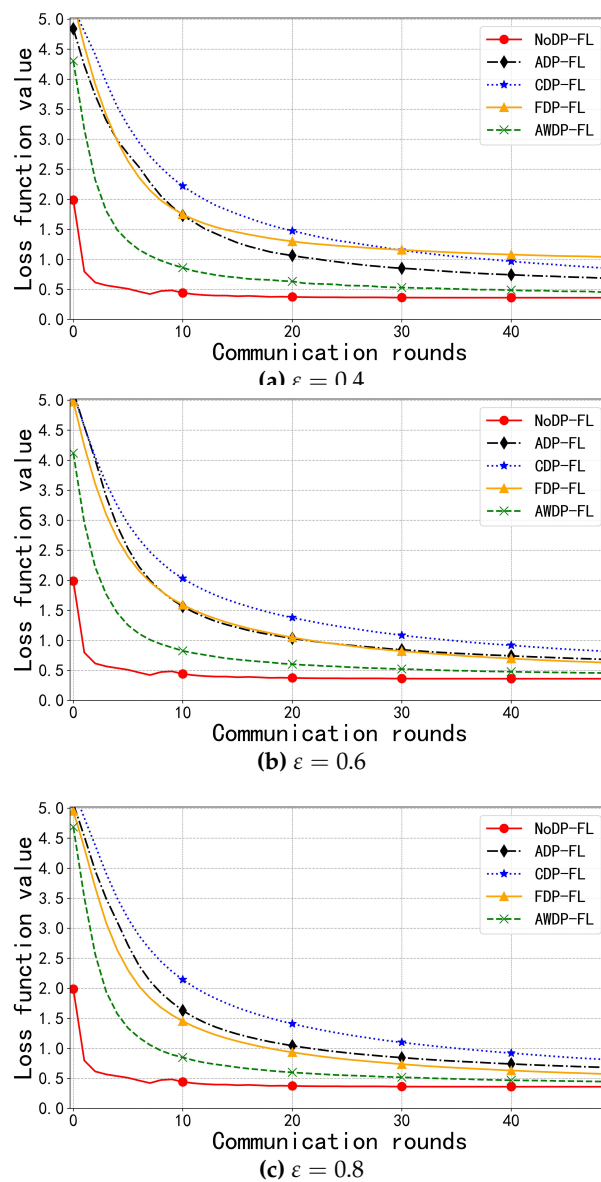


Figure 5. Loss Function Values on the MNIST Dataset

In the MNIST dataset, as shown in Figure 5a-5c, the number of communication rounds is set to 50. From the performance of the NoDP-FL algorithm, it can be seen that without noise perturbation, the model's loss value can quickly find the optimal solution through the optimizer, ultimately converging within 20 rounds. In contrast, the final loss value of NoDP-FL is 0.3601, while our proposed algorithm, after adaptive clipping, even with noise perturbation, still achieves a faster convergence rate in the early stages compared to other algorithms, finally converging within 40 to 50 rounds with a loss value of 0.4323. The loss values for other algorithms such as FDP-FL [17], ADP-FL [18], and CDP-FL [16] are 0.5644, 0.6686, and 0.7994, respectively. It can be seen that the AWDP-FL algorithm has a lower loss value, with a maximum difference of 0.3671. This is mainly due to the algorithm's selective training mode, which allows it to converge in more complex environments and exhibit stronger robustness.

In the Fashion-MNIST dataset, as shown in Figure 6a-6c, we observed that these algorithms were able to converge quickly, and their performance in terms of model accuracy was very similar. This is because the dataset consists of fashion items, with most images being symmetrical, and the unprocessed training set makes it difficult for the algorithms to achieve significant performance improvements compared to the baseline NoDP-FL algorithm. With a privacy budget of 0.8, we evaluated the usability of each algorithm by comparing both model accuracy and loss function values. The model loss value

for NoDP-FL was 0.4395, while the loss value for AWDP-FL was 0.5013. In comparison, the loss values for the other algorithms, FDP-FL [17], ADP-FL [18], and CDP-FL [16], were 0.5959, 0.5976, and 0.6540, respectively. Our AWDP-FL algorithm demonstrated a greater advantage in terms of loss function value, reducing the loss by up to 0.1527, and leading in multiple metrics.

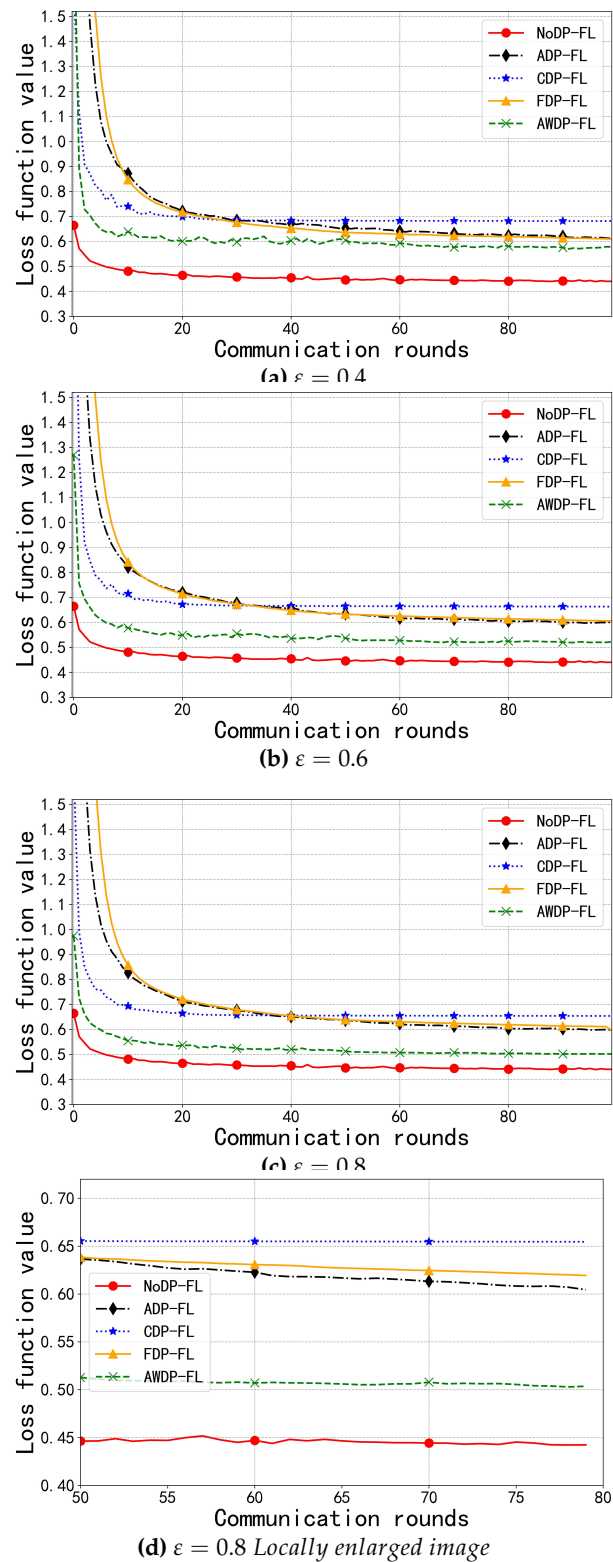


Figure 6. Function Values on the Fashion-MNIST Dataset

For the analysis of the convergence interval, as shown in Figure 6d, with a privacy budget of 0.8, we zoomed in on the interval between 50 and 80 rounds. It can be seen that the AWDP-FL algorithm converged within 50 to 60 rounds, while the other algorithms (except for the CDP algorithm) did not converge within this range. This further indicates that the AWDP-FL algorithm has better robustness while providing privacy protection.

In the CIFAR-10 dataset, as shown in Figure 7a-7c, compared to the Fashion-MNIST dataset, we observed that the distribution of loss function values was more sparse. This is due to the higher complexity of the CIFAR-10 dataset and the relatively larger privacy budget. With a privacy budget of 6, the AWDP-FL algorithm showed lower loss function values during the 5 to 20 rounds of training, indicating that at this stage, the AWDP-FL algorithm not only had stronger performance but also better privacy protection. However, as the noise perturbation increased, the loss function values of AWDP-FL gradually became higher than those of NoDP-FL, which aligns with the basic principles of differential privacy. Ultimately, the loss value difference between AWDP-FL and NoDP-FL was 0.1162. Compared to the other algorithms, AWDP-FL consistently maintained lower loss function values in the CIFAR-10 dataset, with the maximum difference reaching 0.6032 and the minimum difference being 0.1727. This further demonstrates the strong adaptability and usability of the proposed algorithm in complex datasets. However, since the privacy budget in the current experiments is set relatively high, future research should explore the algorithm's performance under lower privacy budgets to ensure comprehensive privacy protection.

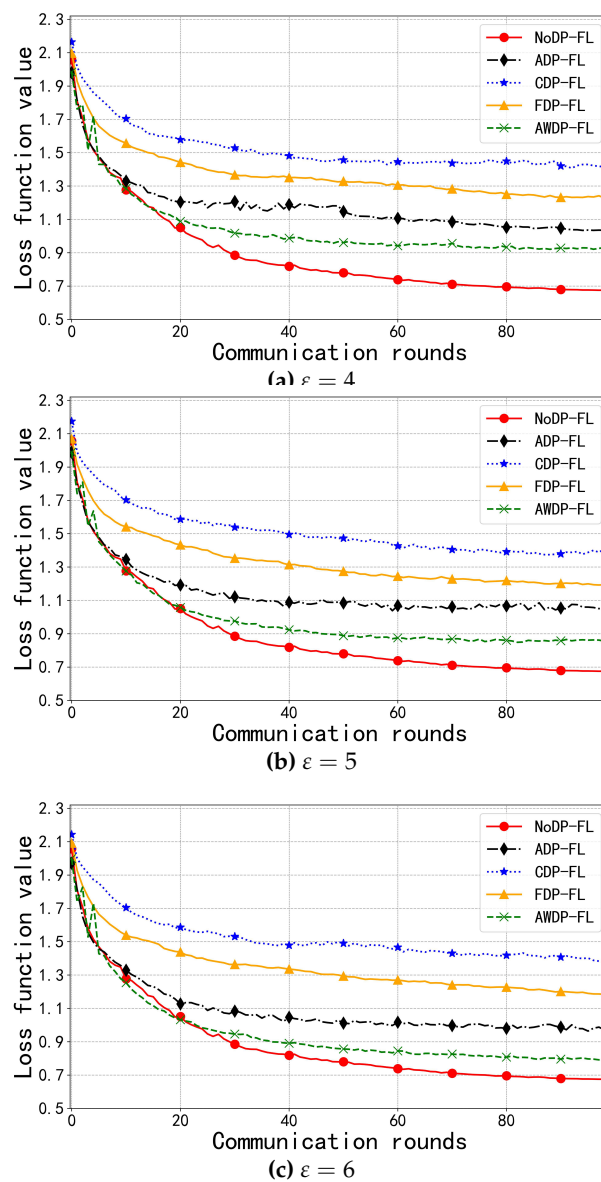


Figure 7. Loss Function Values on the CIFAR-10 Dataset

5. Conclusion

The authors focused on the application of differential privacy in federated learning, particularly in the selection of adaptive clipping thresholds during gradient clipping and the adaptive updating of model gradients. By combining adaptive weight parameters and a hierarchical approach, an Adaptive Differential Privacy Federated Learning framework (AWDP-FL) was designed, with dynamic noise perturbation added before uploading the model parameters. Ultimately, through experimental results on the MNIST, Fashion-MNIST, and CIFAR-10 public datasets, compared with other algorithms, AWDP-FL not only demonstrated excellent performance in providing strong privacy protection but also improved model performance, making the training process more stable.

The privacy budgets used in this paper were evenly distributed. Although smaller privacy budgets were used as much as possible, in some datasets, the privacy budget might still be relatively high, leading to unnecessary privacy budget waste. Future work will explore more precise methods of measuring privacy loss to achieve accurate control of privacy budget consumption. Additionally, since the data sampling methods of clients in a federated learning environment differ, leading to data heterogeneity, subsequent research will focus on how to more effectively implement differential privacy strategies in heterogeneous federated learning scenarios and complex network structures.

Author Contributions: Conceptualization, Z.C. and H.Z.; methodology, Z.C. and H.Z.; validation, H.Z.; investigation, Z.C.; data curation, Z.C. and G.L.; writing-original draft preparation, Z.C.; writing-review and editing, Z.C. and H.Z.; supervision, G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Science and Technology Research Planning Project of the School of Computer Science and Engineering, Changchun University of Technology, Changchun, China, under Grant JJKH20240860KJ.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest

References

- McMahan, B.; Moore, E.; Ramage, D.; et al. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*; PMLR: 2017; pp. 1273–1282. [\[CrossRef\]](#)
- Ma, J.; Naas, S. A.; Sigg, S.; et al. Privacy-preserving federated learning based on multi-key homomorphic encryption. *International Journal of Intelligent Systems* **2022**, *37*(9), 5880–5901. [\[CrossRef\]](#)
- Park, J.; Lim, H. Privacy-preserving federated learning using homomorphic encryption. *Applied Sciences* **2022**, *12*(2), 734. [\[CrossRef\]](#)
- Warnat-Herresthal, S.; Schultze, H.; Shastry, K. L.; et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature* **2021**, *594*(7862), 265–270. [\[CrossRef\]](#)
- Hosseini, S. M.; Sikaroudi, M.; Babaei, M.; et al. Cluster based secure multi-party computation in federated learning for histopathology images. In *Proceedings of the International Workshop on Distributed, Collaborative, and Federated Learning*; Springer Nature Switzerland: Cham, Switzerland, 2022; pp. 110–118. [\[CrossRef\]](#)
- Kanagavelu, R.; Wei, Q.; Li, Z.; et al. CE-Fed: Communication efficient multi-party computation enabled federated learning. *Array* **2022**, *15*, 100207. [\[CrossRef\]](#)
- Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. *Advances in Neural Information Processing Systems* **2019**, *32*. [\[CrossRef\]](#)
- Park, J.; Lim, H. Privacy-preserving federated learning using homomorphic encryption. *Applied Sciences* **2022**, *12*(2), 734. [\[CrossRef\]](#)
- Sun, L.; Lyu, L. Federated model distillation with noise-free differential privacy. *arXiv preprint arXiv:2009.05537* **2020**. [\[CrossRef\]](#)
- Chamikara, M. A. P.; Liu, D.; Camtepe, S.; et al. Local differential privacy for federated learning. *arXiv preprint arXiv:2202.06053* **2022**. [\[CrossRef\]](#)
- Truex, S.; Liu, L.; Chow, K. H.; et al. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*; 2020; pp. 61–66. [\[CrossRef\]](#)
- Sun, L.; Qian, J.; Chen, X. LDP-FL: Practical private aggregation in federated learning with local differential privacy. *arXiv preprint arXiv:2007.15789* **2020**. [\[CrossRef\]](#)
- Zhao, Y.; Zhao, J.; Yang, M.; et al. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal* **2020**, *8*(11), 8836–8853. [\[CrossRef\]](#)
- Liu, W.; Cheng, J.; Wang, X.; et al. Hybrid differential privacy based federated learning for Internet of Things. *Journal of Systems Architecture* **2022**, *124*, 102418. [\[CrossRef\]](#)
- Shen, X.; Liu, Y.; Zhang, Z. Performance-enhanced federated learning with differential privacy for internet of things. *IEEE Internet of Things Journal* **2022**, *9*(23), 24079–24094. [\[CrossRef\]](#)
- Geyer, R. C.; Klein, T.; Nabi, M. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* **2017**. [\[CrossRef\]](#)
- Wu, X.; Zhang, Y.; Shi, M.; et al. An adaptive federated learning scheme with differential privacy preserving. *Future Generation Computer Systems* **2022**, *127*, 362–372. [\[CrossRef\]](#)
- Wang, F.; Xie, M.; Li, Q.; Wang, C. An Adaptive Clipping Differential Privacy Federated Learning Framework. *Journal of Xidian University* **2023**, *(04)*, 111–120. [\[CrossRef\]](#)
- Zhao, J.; Yang, M.; Zhang, R.; et al. Privacy-enhanced federated learning: A restrictively self-sampled and data-perturbed local differential privacy method. *Electronics* **2022**, *11*(23), 4007. [\[CrossRef\]](#)

20. Shen, X.; Liu, Y.; Zhang, Z. Performance-enhanced federated learning with differential privacy for internet of things. *IEEE Internet of Things Journal* **2022**, *9*(23), 24079–24094. [\[CrossRef\]](#)
21. Lian, Z.; Wang, W.; Huang, H.; et al. Layer-based communication-efficient federated learning with privacy preservation. *IEICE TRANSACTIONS on Information and Systems* **2022**, *105*(2), 256–263. [\[CrossRef\]](#)
22. Baek, C.; Kim, S.; Nam, D.; et al. Enhancing differential privacy for federated learning at scale. *IEEE Access* **2021**, *9*, 148090–148103. [\[CrossRef\]](#)
23. Yang, Q.; Liu, Y.; Chen, T.; et al. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* **2019**, *10*(2), 1–19. [\[CrossRef\]](#)
24. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* **2014**, *9*(3–4), 211–407. [\[CrossRef\]](#)
25. Dwork, C.; Rothblum, G. N.; Vadhan, S. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*; IEEE: 2010; pp. 51–60. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.