*electronics*

MDPI

*Article*

# PerFreezeClip: Personalized Federated Learning Based on Adaptive Clipping

**Jianfei Zhang * and Zhilin Liu**

School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China; 2022101129@mails.cust.edu.cn
* Correspondence: jfzhang@cust.edu.cn

**Abstract:** The problem of data heterogeneity is one of the main challenges facing federated learning (FL). Non-IID data usually introduce bias in the training process of FL models, which can impact the accuracy and convergence speed of the models. To this end, we propose a personalized federated learning (PFL) algorithm with adaptive dynamic adjustment of the gradient clipping boundaries and the idea of freezing to reduce the influence of non-IID data on the model, called PerFreezeClip. PerFreezeClip is a design decision regarding parameter architecture, comparing the private and federated models. PerFreezeClip facilitates the training of each device based on an adaptive clipping gradient during training, with more rational updates and more stable gradients. The results based on the CIFAR-10 and CIFAR-100 datasets show that the proposed PerFreezeClip algorithm provides higher test accuracy after controlling the gradient: a maximum of a 50% enhancement compared to typical federated learning (non-personalized) algorithms.

**Keywords:** freezing; gradient tailoring; personalized federated learning



**Citation:** Zhang, J.; Liu, Z. PerFreezeClip: Personalized Federated Learning Based on Adaptive Clipping. *Electronics* **2024**, *13*, 2739. https://doi.org/10.3390/electronics13142739

Academic Editor: Simeone Marino

Received: 31 May 2024
Revised: 9 July 2024
Accepted: 10 July 2024
Published: 12 July 2024



## 1. Introduction

With the rapid development of the big data era, the networked devices in modern distributed networks generate abundant data every day. Research related to deep learning has also experienced explosive growth due to the massive amount of high-quality data samples. Making full use of these highly expressive data can help to construct more complex and accurate machine learning models. However, in practical applications, the issue of data privacy protection is involved. In the process of centralized learning, data need to be uploaded to cloud servers or data centers, which may result in unauthorized access, theft, and leakage of data. In addition, some organizations and individuals with data security protection needs, such as governments and hospitals, may not be able to accept uploading data to a shared platform such as a public cloud, thus restricting their power.

Spawned by the aforementioned issues, researchers have started to gradually shift their focus from data aggregation to model aggregation. Storing data locally and pushing network computation to the edge are becoming increasingly attractive. Federated learning (FL) is a distributed artificial intelligence framework that enables multiple edge devices (such as mobile phones and wearables) to collaboratively train a shared model. In the FL algorithm, edge devices complete the model training process by coordinating with a central server [1,2]. Federated learning also provides valuable insights and potential solutions to data privacy and security challenges in the rapidly evolving field of smart UAV delivery systems [3]. Federated learning addresses the concern of transmitting private information, enables multiple parties to participate in training while protecting data privacy, and solves the problem of data silos.

However, in a federated learning setup, the data are distributed unevenly across the edge devices, resulting in an uneven distribution of data samples. In particular, the local dataset used for training by each edge device is not only different in size but also may contain non-IID data samples. This means there is a data heterogeneity problem, which

can lead to a decrease in the accuracy of the model obtained from the federated learning training, or even cause the training to fail in reaching convergence [4].

Within the IID case, it is observed that the difference between the local scatter weight, which indicates each participant's contribution to the global model aggregation, and the average scatter weight of the central server is minimal. However, in the non-IID case, the gap between the local scatter weight of a client and the average scatter weight of the central server widens with the number of iterations due to data distribution issues.

To address the above problems, we propose identifying a shared parameter space for every client model. By imposing restrictions on the gradient during the client model training and complementing it with this shared parameter space, we can effectively minimize the mutual interference among client models. In this work, our contributions are summarized as follows:

- To address the impact of data heterogeneity on model performance in federated learning, we propose a personalized federated learning method called PerFreezeClip. PerFreezeClip employs freezing and gradient clipping methods to parallelize training and local adaption on the client side, effectively resolving mutual interference among client models.
- We investigate the use of freezing methods to control knowledge transfer between different devices in client-side training and show that by freezing the parameters of certain sub-networks, it is possible to limit the sphere of influence of specific sub-networks, preventing the over-dependence on information from other devices, and thus enabling more accurate knowledge sharing and maintaining localized features (i.e., personalization)
- We investigate limiting the updating of the gradient in the face of data heterogeneity and show that by limiting the updating range of the gradient, we can balance the updating of the weights of global and local information, control the model bias, maintain the consistency of the globally shared parameters, and improve the generalization ability of the model.
- Simulation experiments and performance evaluations of the PerFreezeClip method were conducted on multiple datasets. The experimental results show that PerFreezeClip outperforms personalized methods like FedRep on CIFAR10 and CIFAR100 datasets.

## 2. Related Work

### *2.1. Mitigating Client Drift*

The objective of traditional federated learning is to train a global model that applies to all distributed data without the need for the agent to disclose extensive local information [5]. However, since federated learning [6] aims to achieve high-quality global models by learning from local data of all participating clients in a distributed manner, it overlooks capturing individual device-specific information, resulting in reduced performance in inference and classification. The drop in accuracy is most pronounced when learning on non-independently identically distributed (non-IID) data [7]. This is due to the presence of client-side drift [8]. To address the drift problem caused by processing non-IID data, some methods attempt to enhance local training. Tian et al. propose to add a proximal term to the goal, a term that provides a principled way for servers to address the heterogeneity associated with partial information [6]. Among them, Acar et al. proposed dynamically adjusting the weights of the regularization terms according to the characteristics and contribution of the data on different devices [9]. Similarly, Karimireddy et al. introduced stochastic control averaging, which utilizes a network of controllers to monitor the performance of the global model and performs a weighted average based on the performance. This dynamic process adjusts the contributions of the participants [8]. Li et al. introduced the concept of contrastive loss, which promotes the model to maintain good performance after local updating by maximizing the similarity between pairs of similar samples and minimizing the distance between pairs of dissimilar samples [10]. Kirkpatrick et al. also proposed a combination of Elastic Weight Consolidation (EWC) and Generative Adversarial Network (GAN) approaches to solve the catastrophic forgetting problem [11]. These works focus on calibrating the optimization direction of the local models to narrow the gap between the average model and the global optimum, aiming

to achieve a more stable global model of the device. From an optimization perspective, Wang et al. introduced a new intermediate layer called a "Coordinator" to map and adjust the objective functions of every participant. It enables them to achieve a consistent global optimization goal within a coordinated framework [7].

### *2.2. Personalized Federal Learning*

While the aforementioned studies focus on the performance of training globally shared FL models on heterogeneous data, simply minimizing the average local loss can result in poorer performance, especially when dealing with data heterogeneity. Hence, personalized strategies [12,13] emerged. The main approaches include model mixing, multi-task learning, and local fine-tuning, all of which aim to enhance the performance of subsequent personalization on local data via enhancing the performance of the global model in the presence of data heterogeneity. Personalization techniques are divided into architecture-based and similarity-based approaches.

Architecture-based approaches aim to provide a personalized model architecture tailored to each customer. Among them, Arivazhagan et al. introduced the concept of a personalization layer, which enables the model to be adapted to individual characteristics and needs to achieve the objective of personalized learning [14]. Bui et al. proposed the concepts of a user embedding layer and an aggregation layer to achieve a global user representation by learning user representation in each participant and integrating user embedding vectors from different participants while preserving privacy [15]. There is also a dual aggregation strategy proposed by Liang et al. that consists of two steps: local model updating and global model aggregation [16].

Similarity-based approaches aim to leverage customer relationships to improve the performance of the personalization model. Among them, Smith et al. designed a federated multi-task learning framework that employs a strategy of inter-task sharing and intra-task personalization to enhance personalization performance [17]. Similarly, Corinzia and Beuret proposed an approach based on variational inference. In this approach, each participant is accountable for training their local model and learning the parameters of the global model, along with the task-specific parameters through variational inference [18]. Huang et al. proposed the concept of personalization across data sources to adapt to the characteristics of different data sources through personalized parameter sharing and migration [19]. Collins et al. proposed to improve the performance and generalizability of the model through a shared representation, which attempts to extract common features and knowledge by sharing certain parts of the model [20].

In this paper, we apply the freezing method in local updating and global aggregation and combine it with adaptive gradient clipping to address the data heterogeneity problem. Compared to other personalization methods, the application of freezing methods allows for the better management and optimization of model-specific parameters in local optimization, and since adaptive clipping individualizes the parameters for each device or data source, it allows the personalized model to better fit the local dataset distributions, resulting in a better performance of the model. Table 1 illustrates the relevant work described above.

**Table 1.** Related work.

| Aspect | PerFreezeClip | Personalized Federal Learning | Traditional Federal Learning |
| --- | --- | --- | --- |
| Focus | Personalized federated learning on heterogeneous data | Personalized federated learning on heterogeneous data | Global training of shared FL models on heterogeneous data |
| Main Strategies | Freezing method, adaptive gradient clipping | Model mixing, multi-task learning, local fine-tuning | Limit local updates [6–11] |
| Personalization Approach | Architecture-based | architecture-based [14–16], similarity-based [17–20] | Null |
| Specific Techniques | Freezing in local updating and global aggregation, clipping in local updating | User embedding [15], aggregation layer [14], dual aggregation [16], variational inference [18], parameter sharing [17,19], shared representation [20] | Proximal term [6] |

## 3. The Principle of Perfreezeclip

This paper introduces PerFreezeClip, a novel method aimed at mitigating the impact of non-IID (non-identically independently distributed) data on federated learning (FL) models. Specifically, PerFreezeClip incorporates adaptive clipping (adapt_clip) to address this challenge. The workflow of PerFreezeClip is illustrated in Figure 1. This study employs a freezing method and a gradient clipping strategy to achieve its objectives. PerFreezeClip adopts a gradient clipping technique, although it may incur additional communication overhead [21,22]. However, the introduction of a freezing method for freezing the trunk and head can compensate for the computational overhead incurred in this part. Related work [23,24] has shown that freezing methods can reduce the computational and communication resources required to train learning models in FL. Therefore, this work is not from the perspective of communication overhead but focuses on performance on heterogeneous data. Initially, the global model is distributed from the server to all clients for local training. During local training, the global model is divided into a backbone part and a personalized head. There is usually a division of the model hierarchy, with the backbone part usually containing the bottom and middle layers of the network, which are used to extract generic and universally applicable feature representations, whereas the header is usually located at the top layer of the network or the last layers for a specific task and is used to adaptively tune the model to fit individualized needs. Each client freezes the backbone part and updates the personalized head. Once convergence is achieved, they freeze the head and update the backbone part. Throughout these client model updates, gradient clipping is applied to limit the gradient model. At the end of local training, the client model is passed to the server for aggregation of the backbone parts.

The general form of federal learning is expressed as

$$\min_{x \in \mathcal{R}^d} f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) \tag{1}$$

where $F_i(x) = \mathbb{E}_{\varkappa \sim D_i}[f_i(x, \varkappa)]$ represents the loss function of the $i^{th}$ client, $\varkappa \in \mathbb{Z}$. $D_i$ denotes the data distribution for the $i^{th}$ client. Here, $i \in S$, where $S$ represents a set of clients participating in the training process. For each $i$ and $x$, it is assumed that there is access to an unbiased stochastic gradient $g_i(x)$ of the client's true gradient $\nabla F_i(x)$.

In the context of heterogeneous data, where $i \neq j$, the data distributions $D_i$ and $D_j$ may exhibit notable disparities. Unlike conventional federated learning, our goal is to freeze some of the parameters during the model training and aggregation phases, to improve the stability of the federated learning process, and to obtain a model that is more suitable for individual device customization. Meanwhile, gradient clipping in our algorithm aims at mitigating the effects caused by differences in data distribution and quality across devices, adjusting the performance of the gradient during the training process, and making our algorithm more suitable for each device's characteristics, resulting in distinct models $\theta_i$ for $i \in S$. Drawing upon FedRep [20], we decompose each learning model $\theta_i$ into two components: a global representation model $w_g \in \mathbb{R}^k, \mathbb{R}^d \to \mathbb{R}^k$, which maps data points to a lower-dimensional space of size k, and a device-specific head model $\varphi_i \in \mathbb{R}^p$, where $k + p = d$. Here, $k$ and $p$ denote the dimensions of the global representation and local head models, respectively. To facilitate personalized learning on each device, we employ a strategy where $w_g$ and $\varphi_i$ are frozen in different stages during the training process, and their parameters cannot be updated after freezing.

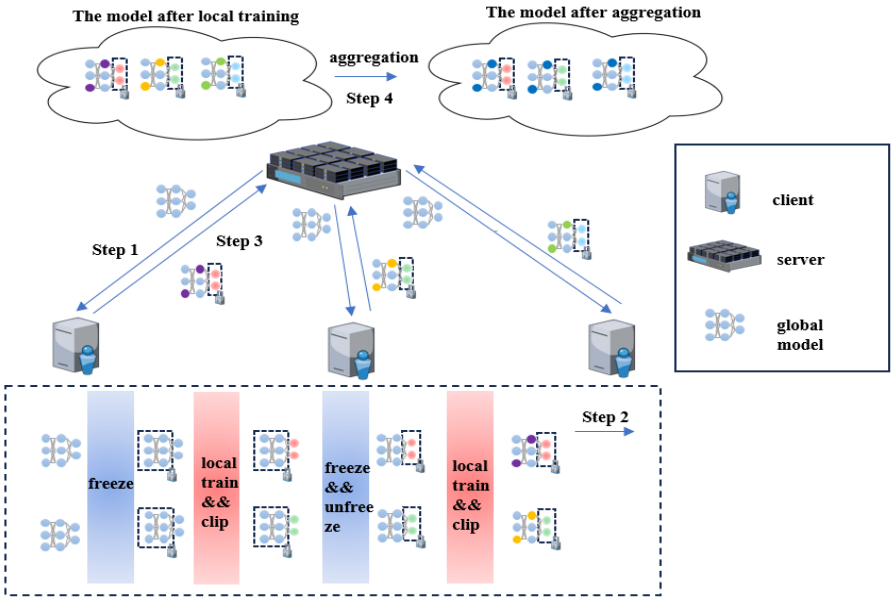


**Figure 1.** PerFreezeClip method flowchart. Step 1: the server sends the global model to the client. Step 2: during the local training process of the client, the model is divided into a backbone part and a personalized head. The backbone part is then frozen, while personalization updates are applied to the head. Additionally, adaptive gradient clipping is employed during the training process to optimize specific segments of the model update. Once the head reaches convergence, it freezes the head and updates the backbone part. In Step 3, after local training, the head is frozen, and the updated global model is returned to the server. The server then aggregates the backbone parts across devices to obtain a new global model.

### *3.1. The Freezing Procedure*

Algorithm 1 illustrates the entire freezing procedure. Suppose that during the t-th round of training, $\theta_i^t$ represents the model on the i-th client, where $i \in S$. Initially, $\theta_i^t$ was received from the global model, $\theta_i^t = \theta_0$. The decoupling process is represented as $\theta_i^t = w_{i,g}^t + \varphi_i^t$, where $w_{i,g}^t$ denotes the decoupled global representation model and $\varphi_i^t$ denotes the personalized head model. The freezing of $w_{i,g}^t$ and $\varphi_i^t$ is managed in different stages in the training process based on the freezing scale parameter $\tau_{w_g}^*$, as indicated by the following equation:

$$\hat{\theta_i^t} \leftarrow Freeze\left(\theta_i^t, L_{w_{i,g}}^t, L_{\varphi_i}^t, e, \tau_{w_g}^*\right) \tag{2}$$

where $\theta_{i,l}^{\hat{t}}$ denotes the model parameters after the freeze is completed. Here, $\theta_i^t$ represents the model of the $i$-th client in t rounds of training. The parameter $\tau_{w_g}^*$ denotes the freezing scale parameter set for the global representation $w_{i,g}^t$, serving as a hyperparameter. Meanwhile, $L_{w_{i,g}}^t$ and $L_{\varphi_i}^t$ signify the respective numbers of layers before the freezing of $w_{i,g}^t$ (global representation) and $\varphi_i^t$ (personalized head). The freezing of $w_{i,g}^t$ and $\varphi_i^t$ is regulated throughout various local epochs in accordance with $\tau_{w_g}^*$. In lines 1–4 of Algorithm 1, in the beginning epoch of local training, it is first determined whether to freeze $w_{i,g}^t$. If the last epoch of freezing $w_{i,g}^t$ is reached, then the unfreezing of $w_{i,g}^t$ is performed, as indicated in lines 11–13. At this point, the transition is made to freezing the head $\varphi_i^t$, indicated in lines 7–10.

The goal of freezing $w_{i,g}^t$ implemented in the local training process is to maintain the static attributes of these layer parameters. This strategy preserves the feature capabilities acquired from prior training, while exclusively updating the weights of the head layer (personalized parameters). As a result, this Freeze method enhances the exploration of the optimization space for the remaining layer parameters during subsequent training, that is, the header layer parameter, thereby fostering personalized learning.

$$\theta_{i,l}^{\hat{t}} = \varphi_{i,l}^{t} + W_{i,g,l} \tag{3}$$

**Algorithm 1** *Freeze*

**Input**: $\theta_i^t, L_{w_{i,g}}^t, L_{\varphi_i}^t, e, \tau_{w_g}^*$
**Output**: $\hat{\theta_i^t}$ //Freeze completed
**if** $e < \tau_{w_g}^* * local\_epoch$:// local_epoch denotes the number of rounds of local training
  **for** $l$ in $L_{w_{i,g}}^t$ **do**:
    final $W_{i,g,l} = w_{i,g,l}^t$; //$W_{i,g}$ denotes an immutable set of parameters
    set $\theta_{i,l}^t = \varphi_{i,l}^t + W_{i,g,l}$
  **end for**
**else**
  **for** $l$ in $L_{\varphi_i}^t$ do:
    final $\Phi_{i,l} = \varphi_{i,l}^t$; //$\Phi_{i,l}$ denotes an immutable set of parameters
    set $\theta_{i,l}^t = w_{i,g,l}^t + \Phi_{i,l}$
  **end for**
**if** $e = \tau_{w_g}^* * local\_epoch$:
  set $\theta_i^t = w_{i,g}^t + \varphi_i$
$\hat{\theta_i^t} = \theta_i^t$//Freeze completed
**end**

The process of freezing $\varphi_i^t$ during global aggregation serves to facilitate knowledge exchange among clients independent of personalized parameters:

$$\theta_{i,l}^{\hat{t}} = w_{i,g,l}^t + \Phi_{i,l} \tag{4}$$

In this context, $W_{i,g}$ and $\Phi_{i,l}$ represent a fixed set of parameters, where $w_{i,g,l}^t \in W_{i,g}, l \in L_{w_{i,g}}^t$ and $\varphi_{i,l}^t \in \Phi_{i,l}, l \in L_{\varphi_i}^t$. This signifies the immutable nature of the parameters, indicating that they are frozen and not subject to updates. Note that it is common to assign a higher value to $\tau_{w_g}^*$. Specifically, in cases involving heterogeneous data with distinct data distributions, this configuration leads to a reduced percentage of freezing for $\varphi_i^t$, allowing for increased local adaptive updates within the head.

### *3.2. Detailed Procedure of Adap_Clip*

After completing the freezing process and starting local training, gradient clipping is applied before updating parameters. In contrast to traditional approaches that employ fixed-value gradient clipping methods, we proposed an adaptive gradient clipping algorithm referred to as adap_clip, which is inspired by the methodology outlined in [25]. adap_clip dynamically calculates the clipping threshold based on recent gradient trends, offering improved adaptability to diverse tasks, data, and models, and enhancing the stability of personalized head parameter updates. Distinguishing it from the method proposed in [25], our adap_clip incorporates a hyperparameter $\psi$ to impose a global hard limit, enabling adjustments to the threshold based on the distribution of gradient trends. This adaptive mechanism ensures that the clipping operation aligns with the current gradient conditions. The adap_clip principle is underpinned by the following observations: (1) The distribution of gradient trends commonly exhibits a long-tailed pattern, with the majority of gradients being small and only a small fraction being large. (2) Parameters associated with smaller gradients predominantly contribute to stabilizing model training, while a subset of parameters with larger gradients may introduce instability, oscillation, or divergence during training.

The adap_clip algorithm predicts the variation in gradient in the current iteration by utilizing historical gradient trends to establish a reasonable gradient boundary. The process is outlined as follows:

Compute the L2 norm of the gradient for each iteration and designate T, the percentile of the historical gradient's L2 norm, as the clipping threshold for the current iteration.

$$P_t \leftarrow {}^{1}/_{\mathcal{B}} \Sigma_{x_i \in X_t} \| \, g_t(x_i) \, \|_2 \tag{5}$$

$$C_t = [P_0, P_1, \cdots, P_t]_T, k \geq 0 \tag{6}$$

where $P_t$ represents the L2 norm of the gradient in the $t$-th iteration, while $C_t$ denotes the gradient clipping threshold in the $t$-th iteration. $X_t$ signifies the batch data selected during the $t$-th iteration, $\mathcal{B}$ is the batch size chosen for each iteration, and $T$ corresponds to the clipping threshold selection percentage $T \in [0, 100]$. The first $T$% of the gradient norm is chosen as the clipping threshold $C_t$.

Following the dynamic computation of the clipping threshold $C_t$, a hyperparameter $\psi$ is introduced as a fixed threshold to restrict the maximum value of gradient norms. Any gradient norm that exceeds this threshold is scaled to match the specified norm. The final clipping threshold is determined by the combination of these two thresholds. While $\psi$ imposes a global hard limit, $C_t$ is adjusted based on the distribution of the gradient norm, making the clipping operation more aligned with the current gradient scenario.

$$C_t = min(C_t, \psi) \tag{7}$$

This approach ensures that the clipping threshold aligns with the gradient norm, thereby effectively managing the gradient magnitude. When confronted with a large gradient norm, the dynamically calculated clipping threshold is used for clipping. Conversely, in scenarios characterized by small gradient norms, the gradient norms themselves are used as the clipping threshold, providing enhanced flexibility to accommodate diverse gradient clipping needs.

The gradient of each data point within every training batch will be subject to clipping. The averaged clipped values of all gradients in that batch will represent the gradient for that iteration.

$$\overline{g_t} = \frac{1}{\mathcal{B}} \sum_{x_i \in X_t} \left\{ g_t(x_i) / max\left(1, \frac{\|g_t(x_i)\|_2}{C_t}\right) \right\} \tag{8}$$

By leveraging the adap_clip algorithm to flexibly determine the clipping threshold, our PerFreezeClip algorithm can dynamically execute gradient clipping for various models and tasks, thereby enhancing model stability and generalization. This adaptive approach offers greater flexibility in addressing diverse gradient distribution scenarios, as opposed to relying solely on a fixed threshold.

### *3.3. Pseudocode for PerFreezeClip*

Initially, PerFreezeClip selects a set of clients $S$ from $K$, where $S \in K$ and the sampling. Meanwhile, the server holds model $\theta_0$. PerFreezeClip, as shown in Algorithm 2, consists of two primary steps: local training and server aggregation. It incorporates freezing and clipping methods during the local training phase. In Algorithm 2, lines 5–17, the process of local training for each client is described. In line 6, each local model decides whether to freeze the global representation $w_{i,g}^t$ or the personalized head $\varphi_i^t$ at the current epoch, based on the epoch it is currently in and the freezing scale parameter $\tau_{w_g}^*$. The detailed process of freezing is described in Algorithm 1. The process of updating and cropping is described in lines 9–14, where for each round of local training, adaptive gradient cropping is imposed. Line 19 describes the process of aggregating $w_{i,g}^t$. The update operation for local training is illustrated by Equation (9):

$$x_{k+1,j} = CLENTOPT\left(x_{k,j}, \overline{g}_{t,e}^k\left(x_{k,j}\right), \eta, t\right) \tag{9}$$

The optimizer $CLENTOPT$, representing a gradient-based optimizer (e.g., SGD) with a learning rate of $\eta$, describes the update process for lines 13 and 14 in Algorithm 2. Here, $\overline{g}_{t,e}^k\left(x_{k,j}\right)$ signifies the clipped gradient. Furthermore, beyond the final round encompassing the update of all parameters, the specific parameter updates are contingent upon the frozen state during various epochs of the local training. In each round of local training, a freezing procedure (Algorithm 1) is first entered. At the beginning of training, the trunk will be frozen according to the freezing ratio parameter $\tau_{w_g}^*$. This hyperparameter will determine at how many epochs the backbone will be frozen. At this time, the weights of the head

layer will be updated to allow the head to adapt to the local data quickly. At the same time, in the update of the head layer, gradient clipping will be used to limit the gradient to ensure the stability of the overall gradient. In the frozen state $\theta_{i,l}^{\hat{t}} = \varphi_{i,l}^{t} + W_{i,g,l}$, the parameter updates adhere to Equation (10):

$$\varphi_{t,e+1}^{k} \leftarrow \varphi_{t,e}^{k} - \eta \overline{g}_{t,e}^{k} \tag{10}$$

Conversely, in the state $\theta_{i,l}^{\hat{t}} = w_{i,g,l}^{t} + \Phi_{i,l}$, the updates are governed by Equation (11):

$$w_{g,t,e+1}^{k} \leftarrow w_{g,t,e}^{k} - \eta \overline{g}_{t,e}^{k} \tag{11}$$

Global aggregation involves amalgamating all updated models after completing local training, following FedAvg's standard aggregation procedure. At this juncture, $\theta_i^t = w_{i,g}^t + \Phi_i$, wherein $\theta_i^t$ signifies the locally trained model for each client $i \in S$, and $\theta_{t+1}$ denotes the global model for the next iteration. The process is summarized in Algorithms 1 and 2.

$$\theta_{t+1} = \frac{1}{|S|}\sum_{i \in S} \theta_i^t = \frac{1}{|S|}\sum_{i \in S} w_{i,g}^t \tag{12}$$

**Algorithm 2** *PerFreezeClip*

**Input**: Initialize model parameters $\theta_0$, learning rate $\eta$, client set S, number of clients involved in model training $K$, number of communication rounds $T$, number of local iterations $E$, size of local training batch B, percentile of gradient clipping threshold selection T, freezing ratio $\tau^*$ for global representation of $w_g$, maximal value of norms $\psi$

**for** $t = 1, 2, \cdots, T$ **do**
  server Send $\theta_0$ to all clients
  **for** each client $i \in S$ in parallel **do**
    $\theta_i^t = \theta_0$;//initialization
    **for** $e = 1, 2, \cdots, E$ **do**// Start training
      $\hat{\theta_i^t} \leftarrow Freeze\left(\theta_i^t, L_{w_{i,g}}^t, L_{\varphi_i}^t, e, \tau_{w_g}^*\right)$ //denote the number of $w_g$ and $\varphi_i$ layers before freezing
      **for** $I_i^k \in D_k$ **do**// $I_i^k$ is a randomized non-repeating batch of data from $D_k$
        **for** $x_{k,j} \in I_i^k$ **do**
          $g_{t,e}^k(x_{k,j}) \leftarrow \nabla F_{t,e}^k(x_{k,j})$; //Calculate the gradient
          $C_e^k = \left[P_0^k, P_1^k, \cdots, P_n^k\right]_T$;//Adaptive clipping threshold selection
          $C_e^k = \min(C_e^k, \psi)$
          $\overline{g}_{t,e}^k(x_{k,j}) \leftarrow \frac{1}{\mathcal{B}}\Sigma_{x_{k,j} \in I_i^k}\left\{g_{t,e}^k(x_{k,j}) / \max\left(1, \frac{\|g_t(x_{k,j})\|_2}{C_e^k}\right)\right\}$
          $w_{g,t,e+1}^k \leftarrow w_{g,t,e}^k - \eta \overline{g}_{t,e}^k$; //update
          $\varphi_{t,e+1}^k \leftarrow \varphi_{t,e}^k - \eta \overline{g}_{t,e}^k$; //update
        **end for**
      **end for**
    **end for**
  **end for**
$\theta_{t+1} = \frac{1}{|S|}\sum_{i \in S} \theta_i^t = \frac{1}{|S|}\sum_{i \in S}\left(w_{i,g}^t, \varphi_i^t\right) = \frac{1}{|S|}\sum_{i \in S} w_{i,g}^t$//$\varphi_i^t$ freezing state
Randomly select a subset of clients $S_{t+1}$ without repetition and send the updated model $\theta_{t+1}$ to all clients in this subset
**end for**

## 4. Theoretical Analysis of Gradient Clipping

In heterogeneous data environments, we contend that during each communication round involving personalized learning on devices, the local model is updated towards the local optimum, which may diverge significantly from the global optimum. As a result, the average model may also deviate from the global optimum, especially in scenarios with significant local updates. These updates actually move along the gradient direction. Thus, a straightforward method to regulate these updates is by restricting the gradient, thereby constraining the gradient range within the personalized context. In addressing this issue, we counteract overfitting in local updates by applying a gradient clipping technique during parameter updates to confine the gradient range. The subsequent analysis is detailed below:

During the aggregation step, $\varphi_i^t$ remains in a frozen state, indicated by $\varphi_i^t = \Phi_{i,l}$, where $\Phi_{i,l}$ denotes an immutable set of parameters. Notably, $\varphi_i^t$ does not participate in the aggregation update. We can reframe the FedAvg update as follows:

$$\theta_{t+1} = \frac{1}{|S|}\sum_{i\in S}\theta_i^t = \frac{1}{|S|}\sum_{i\in S}\left(w_{i,g}^t, \Phi_{i,l}\right) = \frac{1}{|S|}\sum_{i\in S} w_{i,g}^t \tag{13}$$

When it is evident that $\varphi_i^t$ in $\theta_i^t$ consistently remains uninvolved in the update, it can be interpreted as $\theta_i^t = w_{i,g}^t$. Another reformulation of Equation (13) is presented below:

$$\theta_{t+1} = \frac{1}{|S|}\sum_{i\in S}\theta_i^t = \theta_t - \frac{1}{|S|}\sum_{i\in S}\left(\theta_t - \theta_i^t\right) = w_g^t - \frac{1}{|S|}\sum_{i\in S}\left(w_g^t - w_{i,g}^t\right) \tag{14}$$

where we define $\Delta_i^t := w_g^t - w_{i,g}^t$ to represent the update of the $i$-th client. Subsequently, $\Delta_t := (1/|S|)\sum_{i\in S}\Delta_i^t$ is introduced to denote the global update obtained by averaging the updates of the global representation of all clients (i.e., the difference between the global model $\theta_{t+1}$ and the global model $\theta_t$ from the previous round). Given that $\varphi_i^t$ is in a frozen state and does not participate in the update, $\theta_{t+1} = w_g^{t+1} + \varphi_i^{t+1}$ and $\theta_t = w_g^t + \varphi_i^t$. Here, $\varphi_i^{t+1} = \varphi_i^t$. The parameter update can be denoted as $\theta_{t+1} \leftarrow \theta_t$ and $w_g^{t+1} \leftarrow w_g^t$.

Based on the aforementioned definition, it is evident that $\Delta_t$ functions as the pseudo-gradient of the applied stochastic gradient descent (SGD) algorithm, guiding the server's updates to the global representation $w_g^t$. This definition clarifies that after using SGD on the client, additional operations can be conducted to manage the pseudo-gradient $\Delta_t$, including gradient clipping. As a result, we adaptively scale the gradients of the parameters before updating them. Furthermore, we posit the following assumptions:

**Assumption 1** (Lipschitz Gradient) [26]**.** *The function $F_i$ is L-smooth for all $i \in [m]$, i.e., $\| \nabla F_i(x) - \nabla F_i(y) \| \leq L \| x - y \|$, for all $x, y \in \mathbb{R}^d$.*

Assumption 1 presents a restriction on the gradient of the loss function $F_i(x)$ for each client, referred to as the Lipschitz Gradient condition. Let us consider a set of clients, each associated with a loss function $F_i(x)$, where $i \in [m]$ denotes the client index and m represents the total number of clients. The parameter L in this assumption represents a positive constant that denotes the Lipschitz constant of the gradient $\nabla F_i(x)$ for each client. It is postulated that the gradient $\nabla F_i(x)$ of the loss function $F_i(x)$ for each client complies with the following property: for all $x, y \in \mathbb{R}^d$ (where d signifies the dimension of the real number space), the product of the disparity in the norms of the gradient and the distance between the variables x and y does not surpass L. Here, $\| \cdot \|$ denotes the L2 norm (Euclidean norm).

We argue that the gradient of the loss function does not change too drastically for each client, and that the rate of change in the gradient is globally limited to a constant L. This is a smoothing requirement that ensures the gradient of the loss function does not vary too much in the local region, which is beneficial for the stability and convergence of the optimization algorithm. Therefore, the Lipschitz Gradient assumption ensures that PerFreezeClip can find an L (threshold) for limiting the range of gradient updates among several local adaptation steps. It indirectly satisfies the condition of Lipschitz continuity by setting the threshold for gradient clipping. The Lipschitz Gradient (Gradient clipping) ensures that the rate of change in the gradient is somewhat limited, which helps to enhance the convergence speed and stability of PerFreezeClip.

Hence, to correctly apply gradient clipping and satisfy Lipschitz continuity, it is important to wisely choose the threshold for gradient clipping. Setting the threshold too small may cause the gradient to be over-clipped, affecting the convergence performance of the model, while setting the threshold too large may not satisfy the Lipschitz continuity condition. A wise choice of threshold for gradient cropping can be a combination of the following four aspects. (1) Initial estimation based on the model and the task; the

typical range of gradient can be estimated through preliminary experiments or based on previous experience with similar tasks. (2) Based on the statistical analysis of the gradient, the statistical properties of the gradient, including the mean and standard deviation, are monitored during the training process. (3) Assemble model architectures and optimizers, which may have different sensitivities to gradient tailoring. (4) Experimental validation, where a series of experiments are conducted to validate the model's performance after the threshold is set. Therefore, we introduce the hyperparameter $\psi$ for a global hard limit, which is analyzed by the following assumptions:

**Assumption 2** (Bounded Gradients) [27]**.** *The function $f_i(x, \varkappa)$ has G-bounded gradients; i.e., for any $i \in [m], x \in \mathbb{R}^d$, and $\varkappa \in \mathfrak{X}$, we have $\left|[\nabla f_i(x, \varkappa)]_j\right| \leq G$ for all $j \in [d]$.*

Assumption 2 involves a restriction on the gradient of the loss function $f_i(x, \varkappa)$ on each client $i$ and all dimensions $j$ and is known as the Bounded Gradients condition. Specifically, assume that for each client $i$ ($i \in [m]$ denotes the index of the client), the gradient $\nabla f_i(x, \varkappa)$ of the loss function $f_i(x, \varkappa)$ is bounded by the absolute value of the gradient $\nabla f_i(x, \varkappa)$ in all dimensions $j$ for any $x \in \mathbb{R}^d$ and $\varkappa \in \mathfrak{X}$ ($\mathfrak{X}$ denotes the set of some random variable). That is, $\left|[\nabla f_i(x, \varkappa)]_j\right| \leq G$, where G is a positive constant representing the upper bound of the gradient of each client's loss function in all dimensions.

We argue that the gradient of each client's loss function does not exceed G in each dimension. This assumption ensures that the gradient of each client's loss function is not too large in each dimension, constrained by a global upper bound G. The gradient boundedness condition in Assumption 2 can be seen as a formal representation of gradient clipping. The hyperparameter $\psi$ we introduce is the upper bound G on this global restriction. This can help us to better limit the update of the gradient.

## 5. Experiment and Results

### *5.1. Experimental Setup*

#### 5.1.1. Datasets and Model Architectures

This chapter presents experiments conducted on the CIFAR-10 and CIFAR-100 image classification tasks within the context of federated learning. The CIFAR-10 dataset consists of 10 classes of Red–Green–Blue (RGB) color images, each sized at 32 × 32, with 6000 images per class, totaling 50,000 training images and 10,000 test images. On the other hand, the CIFAR-100 dataset consists of 100 classes, with 600 32 × 32 color images per class, where there are 500 images for training and 100 images for testing. To preserve non-IID data partitioning among devices, we allocated distinct classes from the respective CIFAR-10 or CIFAR-100 datasets to each of the 100 client devices. This allocation ensured an even distribution of data samples across the devices.

#### 5.1.2. Implementation Details

Stochastic gradient descent (SGD) with a momentum of 0.5 was utilized in every algorithm that would be compared with our PerFreezeClip in the following simulations. For both CIFAR-10 and CIFAR-100, a local sample batch size of 10 was employed. The client participation rates were set to $\alpha = 0.1$, and all clients were sampled in each round during several local adaptive step phases. The clipping threshold in gradient clipping was dynamically computed based on the gradient norm, with a maximum norm selected in $\psi \in (30, 80)$ to limit clipping, determined empirically. The dynamically computed clipping threshold was chosen based on the gradient norm share, where the top 90% of gradient norms were used as the clipping threshold in the experiment. Furthermore, we analyzed and compared scaling methods for a fixed clipping threshold. For other methods, the learning rate followed FedAvg's default base learning rate of 0.01.

In the freezing setup, local updates of the backbone part of PerFreezeClip and the personalization layer were fine-tuned through freezing operations for each corresponding epoch, based on the parameter $\tau^*_{w_g}$ (indicating the freezing ratio for the global representation of

$w_{i,g}^{t}$), $\tau_{w_g}^{*} = [0.1, 1)$. All models were randomly initialized and trained for 100 communication rounds. PerFreezeClip performs 10 epochs for each local update. The clients and classes were set to (100,2) and (100,5). Other methods utilized the same local batch sample size. We conducted 10 local update iterations for local adaptive fine-tuning of the learned model using local training data samples from each device, and the accuracy was computed by averaging the local accuracies of all users in the last 10 rounds of communication.

#### 5.1.3. Baselines

We compared the performance of our proposed algorithm, PerFreezeClip, with the following algorithms: FedAvg and FedProx are traditional single-model FL approaches designed to address data heterogeneity. FedRep, LG-FedAvg, Ditto, and FedPer utilize personalization strategies to mitigate data heterogeneity.

#### 5.1.4. Performance Metrics

In our investigation, we consider test accuracy as a crucial performance metric. We employed multiple local adaptive fine-tuning steps for the PerFreezeClip and FedRep algorithms to construct a local head model for each device. The models are subsequently subjected to personalized learning through a specific fine-tuning approach. In the local adaptive step of PerFreezeClip, fine-tuning encompasses freezing and clipping operations. The freezing ratio quantifies the epoch when freezing occurs, enabling the assessment of the impact of different freezing ratios on accuracy. Subsequently, relevant experiments are conducted to assess the impact of the freezing ratio and maximum norm settings on accuracy. Moreover, we evaluate and analyze the application of different clipping methods within the framework of PerFreezeClip.

### *5.2. Performance Evaluation*

#### 5.2.1. Performance Comparison with the Baseline

Table 2 presents the test accuracies of PerFreezeClip in comparison to the baseline methods, where model accuracies are represented in percentage values.

For CIFAR-10, when C = 2, denoting a lower data heterogeneity and minimal influence on the model, FedRep achieved an accuracy of 86.65%, while PerFreezeClip demonstrated a 1.84% improvement over FedRep. Furthermore, PerFreezeClip demonstrated enhancements ranging from 1.36% to 4.35% compared to other personalized federated learning algorithms such as LG-Fed, Ditto, and FedPer. It also exhibits improvements of approximately 46.14% to 48.57% in comparison to single models like FedProx and FedAvg. As the data heterogeneity intensified with C = 5, the accuracies of all personalized algorithms decreased. However, PerFreezeClip can still maintain a 1.23% at least (to FedRep) and 14.53% at most (to LG-FedAvg). Although the performance of FedAvg and FedProx increased around 11%, PerFreezeClip also demonstrated enhancements of 25.77% and 26.56%, respectively.

**Table 2.** Tested accuracy of PerFreezeClip compared to the baseline methods on the CIFAR-10 and CIFAR-100 datasets when $\tau_{w_g}^{*} = 0.9$. $\psi$ is the maximum value of norms, C is the class of random samples obtained by the client, and $\tau_{w_g}^{*} = 0.9$ is the most adequate for personalized learning. The values of FedRep in the brackets were obtained from the original paper.

| Dataset | n/S | *PerFreezeClip* | *FedRep* | *LG−Fed* | *Ditto* | *FedPer* | *FedProx* | *FedAvg* |
|---|---|---|---|---|---|---|---|---|
| CIFAR-10 | C = 2 | 88.49 ($\psi$ = 35) | 86.65 (87.70) | 84.14 | 85.39 | 87.13 | 39.92 | 42.65 |
| | C = 5 | 77.55 ($\psi$ = 55) | 76.32 (75.68) | 63.02 | 70.34 | 73.84 | 50.99 | 51.78 |
| CIFAR-100 | C = 5 | 83.66 ($\psi$ = 50) | 83.47 (79.15) | 72.44 | 78.91 | 76.00 | 20.17 | 23.94 |
| | C = 20 | 75.90 ($\psi$ = 35) | 75.09 (56.10) | 38.76 | 56.34 | 55.68 | 28.52 | 31.97 |

For CIFAR-100, PerFreezeClip achieved similar results. It demonstrated accuracy improvements of 0.19% and 0.81% compared to FedRep for C = 5 and C = 20, respectively. But PerFreezeClip exhibits significantly greater enhancements compared to other personalized

algorithms. It maintained 4.67% at least (to Ditto) and 11.22% at most (to LG-FedAvg) with C = 5, and 19.56% at least (to Ditto) and 37.14% at most (to LG-FedAvg) with C = 20. PerFreezeClip also demonstrated enhancements of 43.93% at least and 63.49% at most, respectively. These results are significantly better than those of CIFAR-10.

Conclusively, PerFreezeClip exhibits comparable performance to FedRep and outperforms other baseline algorithms in terms of fine-tuned test accuracy. And PerFreezeClip demonstrates an effective mitigation of the impact on non-IID data compared to the single-model federal approach. In comparing CIDAR-10 and CIDAR-100, the simulations conducted on CIDAR-100 exhibit greater heterogeneity. The performance of FedAvg and FedProx decreased significantly, and other personalized federated learning algorithms also decayed severely. Nonetheless, PerFreezeClip demonstrated consistent performance with minimal degradation, indicating its stability when processing heterogeneous data.

#### 5.2.2. Applying Different Clipping Methods on PerFreezeClip

Figure 2 illustrates the performance of two clipping methods within the PerFreezeClip framework under a scenario emphasizing personalized information. After T = 50 rounds, when personalization is fully learned, the accuracy of FedRep becomes susceptible to overfitting of the personalized head, which results in a decreased generalization ability and slower convergence. In this stage, PerFreezeClip demonstrates a more stable convergence rate than FedRep in both of the value-based and adaptive clipping approaches, resulting in higher accuracy. Specifically, compared to FedRep, PerFreezeClip achieves a 2.38% accuracy improvement using the value-based clipping method and a 1.84% improvement with adaptive clipping.

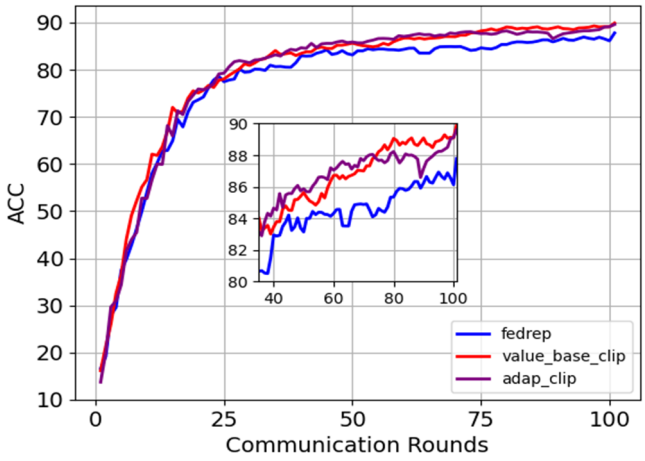


**Figure 2.** Performance comparison of applying two different clipping methods on PerFreezeClip with FedRep on CIFAR-10 dataset when $\tau^*_{w_g}$ = 0.9 and C = 2. value_based_clip is a clipping based on a fixed threshold and adap_clip is a clipping that adaptively selects a threshold based on a gradient norm with a maximum norm of $\psi$ = 35.

Figures 3 and 4 depict the performance results of PerFreezeClip compared to FedRep after fine-tuning maximum norms using different clipping methods. In Figure 3, by using adaptive clipping, PerFreezeClip outperforms FedRep at maximum norms of 35, 40, and 45, exhibiting accuracy improvements of 1.84%, 1.58%, and 0.99%, respectively. Optimal performance is achieved at a maximum norm of 35. Similarly, in Figure 4, by utilizing a fixed-value-based clipping approach, PerFreezeClip achieves higher accuracies than FedRep at thresholds of 35, 40, and 45. The accuracy improvements are 2.38%, 1.79%, and 3.06%, respectively, demonstrating optimal performance at a threshold of 45.

Overall, limiting the gradient through clipping can indirectly influence personalized head updates and mitigate the adverse effects of personalized overlearning. Notably, compared to FedRep, PerFreezeClip exhibits superior performance with either value-based or adaptive clipping based on gradient norms.

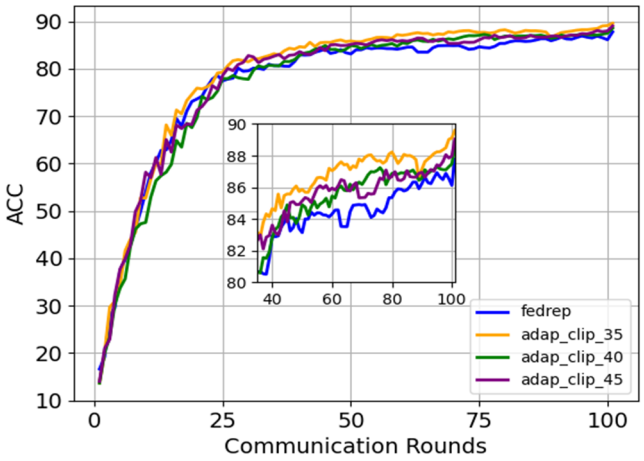


**Figure 3.** Performance comparison of applying adap_clip on PerFreezeClip with FedRep after fine-tuning the maximum value of norms when $\tau^*_{w_g}$ = 0.9 and C = 2, where $\psi$ is taken as 35, 40, and 45.

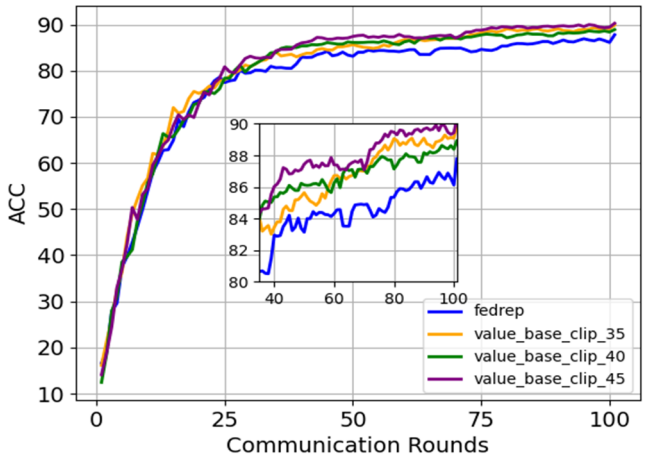


**Figure 4.** Performance comparison of applying value_based_clip on PerFreezeClip with FedRep after fine-tuning the maximum value of norms when $\tau^*_{w_g}$ = 0.9 and C = 2, where $\psi$ is taken as 35, 40, and 45.

#### 5.2.3. Impact of Personalized Fine-Tuning

We investigated the influence of personalized learning on the final test accuracy of PerFreezeClip and FedRep after multiple fine-tuning steps during local training. As the number of local training rounds on each device was set to 10, the degree of personalized learning can be quantified as the number of training rounds on personalized layers during local training, and this number was determined by the freeze ratio. Notably, we do not predetermine the maximum value of norms during the fine-tuning process. Instead, it relies on experiential knowledge to establish the optimal value.

In Table 3, it is evident that both PerFreezeClip and FedRep achieved higher accuracy on CIFAR-10 through personalized fine-tuning. When C = 2, the accuracy of FedRep improved from 86.65% to 91.62%, while PerFreezeClip also increased from 88.49% to 93.61%. In the context of synchronous growth, PerFreezeClip maintained a 1.99% performance advantage over FedRep. Likewise, with C = 5, where data heterogeneity was intensified, the accuracy of FedRep improved from 76.32% to 83.58%, while PerFreezeClip increased from 77.55% to 84.03%. Under a worse case of data heterogeneity, PerFreezeClip remained 0.45% higher than FedRep.

Similar effects on CIFAR-100 are illustrated in Table 4. Personalized fine-tuning can improve the accuracy of models. When C = 5, PerFreezeClip achieves accuracy improvements of 0.27%, 1.36%, and 1.58% over FedRep in each stage of personalized fine-tuning, starting from $\tau^*_{w_g}$ = 0.8. Although enlarging the value of C made the heterogeneity of the

data distribution even worse, when C = 20, PerFreezeClip can enhance its accuracy by 0.36%, 0.14%, and 0.27% over FedRep in each stage of personalized fine-tuning.

**Table 3.** Test accuracy of PerFreezeClip and FedRep after fine-tuning the degree of personalized learning on the CIFAR-10 dataset. $\tau^*_{w_g}$ = 0.9 implies the fullest personalized learning.

| Dataset | n/S | Algorithm | Personalized Fine-Tuning $\tau^*_{w_g}$ | |
|---|---|---|---|---|
| | | | $\tau^*_{w_g}$ = 0.9 | $\tau^*_{w_g}$ = 0.5 |
| CIFAR-10 | C = 2 | *PerFreezeClip* | 88.49 ($\psi$ = 35) | 93.61 ($\psi$ = 55) |
| | | *FedRep* | 86.65 | 91.62 |
| | C = 5 | *PerFreezeClip* | 77.55 ($\psi$ = 55) | 84.03 ($\psi$ = 50) |
| | | *FedRep* | 76.32 | 83.58 |

**Table 4.** Test accuracy of PerFreezeClip and FedRep after fine-tuning the degree of personalized learning on the CIFAR-100 dataset. $\tau^*_{w_g}$ = 0.9 implies the fullest personalized learning.

| Dataset | n/S | Algorithm | Personalized Fine-Tuning $\tau^*_{w_g}$ | | | |
|---|---|---|---|---|---|---|
| | | | $\tau^*_{w_g}$ = 0.9 | $\tau^*_{w_g}$ = 0.8 | $\tau^*_{w_g}$ = 0.7 | $\tau^*_{w_g}$ = 0.5 |
| CIFAR-100 | C = 5 | *PerFreezeClip* | 84.61 ($\psi$ = 80) | 86.17 ($\psi$ = 40) | 88.19 ($\psi$ = 40) | 88.68 ($\psi$ = 40) |
| | | *FedRep* | 83.47 | 85.90 | 86.83 | 87.10 |
| | C = 20 | *PerFreezeClip* | 75.90 ($\psi$ = 35) | 79.25 ($\psi$ = 40) | 80.70 ($\psi$ = 70) | 81.13 ($\psi$ = 80) |
| | | *FedRep* | 75.09 | 78.86 | 80.56 | 80.86 |

In summary, the simulations indicated that personalized models are sensitive to the degree of personalized learning. Maximizing the value of $\tau^*_{w_g}$ would lead to an overfitting problem, as the model overly relies on personalized data on clients and ignores general patterns and trends. Thus, an exaggerated $\tau^*_{w_g}$ reduced the model's generalization ability and accuracy. Subsequently, by mitigating degrees of personalized learning, personalized fine-tuning led to improved accuracy for both PerFreezeClip and FedRep. Meanwhile, PerFreezeClip demonstrated superior performance.

#### 5.2.4. The Effect of the Maximum Norm ψ

We investigated the impact of the PerFreezeClip algorithm on the final test accuracy by limiting the gradient size based on the maximum value of gradient norms $\psi$. Figures 5 and 6 depict the influence of PerFreezeClip on the final test accuracy after fine-tuning the maximum value of norms.

In Figure 5a, setting the maximum value of norms to 35 yields the highest performance for PerFreezeClip. Afterward, the accuracy decreases accordingly. Subsequently, Figure 5b–d demonstrate fluctuations in the final test accuracy of PerFreezeClip. Nonetheless, the best value of $\psi$ among various maximum norm thresholds can still be selected. Across a–d in Figure 5, PerFreezeClip consistently outperforms FedRep when the maximum value of norms is set to 35, 55, 55, and 50, respectively. This emphasizes the significance of selecting optimal thresholds for peak performance.

Similar effects also appear in Figure 6. Pronounced fluctuations in PerFreezeClip can be observed on CIFAR-100 with the adjustment in the maximum value of norms. The accuracy of PerFreezeClip exhibits an upward trend, ultimately identifying the optimal threshold in each case. In Figure 6a,c, the optimal accuracy for PerFreezeClip is achieved with a maximum value of norms of 80 and 35, respectively. In contrast, in Figure 6c, a heightened heterogeneity and maximal personalized learning lead to a decreasing trend in accuracy as the maximum number of norms increases, highlighting the need for more stringent gradient limitations under these conditions to achieve the best accuracy.

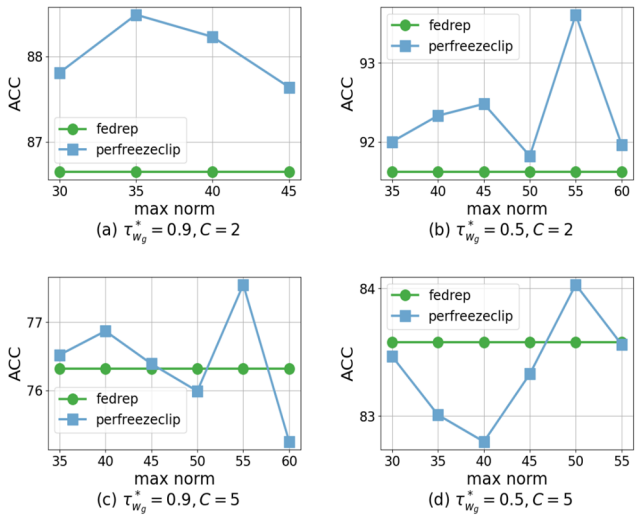


**Figure 5.** Changes in test accuracy of PerFreezeClip after fine-tuning the maximum value of norm on the CIFAR-10 dataset.

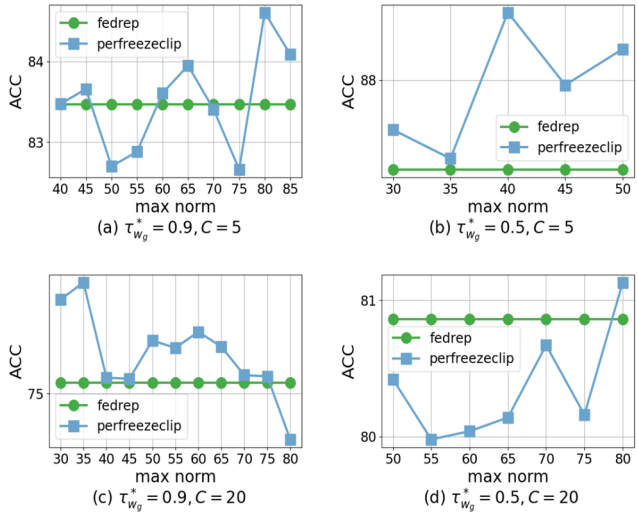


**Figure 6.** Changes in test accuracy of PerFreezeClip after fine-tuning the maximum value of norm on the CIFAR-100 dataset.

This emphasizes the importance of choosing the appropriate $\psi$ tailored to specific problem settings and dynamics. A smaller $\psi$ may excessively restrict the gradient. This may result in the inability to use the threshold obtained from dynamic computation. As a result, PerFreezeClip cannot achieve optimal generalization performance. Conversely, a larger $\psi$ may inadequately control the gradient size. This hindered the effective training of PerFreezeClip. Thus, PerFreezeClip must carefully select the maximum value of norms to achieve optimal performance across diverse tasks and settings.

#### 5.2.5. Effect of the Number of Local Iterations on PerFreezeClip

We investigated the impact of adjusting the number of local iterations on the test accuracy of PerFreezeClip. Given the varying numbers of local epochs under consideration, we refrain from targeting an optimal maximum value of norms across epochs, and instead focus on a generalized empirical analysis. We observe that the test accuracy tends to increase with an escalation in local iterations. We specifically employ the optimal value of $\psi$ when the number of local epochs is set to 10, recognizing that this value may not be universally optimal for other local epochs.

Empirically, a larger number of local epochs entails more gradient updates on the client in each communication round. Figure 7 illustrates the comparison of PerFreezeClip and FedRep with 2, 10, 20, 30, and 40 local epochs. Overall, PerFreezeClip surpasses FedRep in certain epochs. Notably, even when local epochs are set to 2, PerFreezeClip outperforms FedRep when the maximum value of norms approximates the optimal value for that epoch. However, when the number of local epochs is 20 or 30, PerFreezeClip's test accuracy improvement is not significant. When the number of local epochs is 40, the training accuracy decreases, which is not what was previously expected. This suggests that the freezing ratio parameter and the maximum value of norms are inappropriate at this time for a local epoch number of 40.

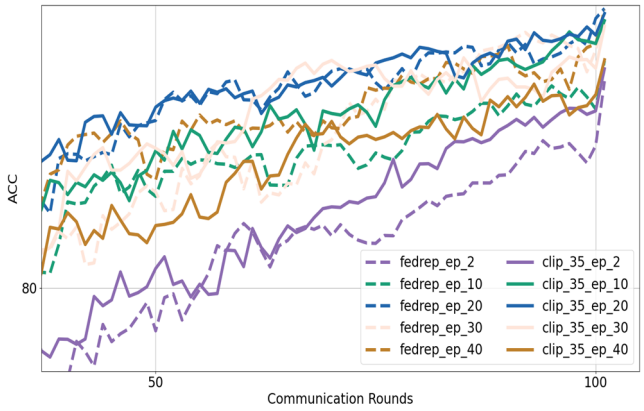


**Figure 7.** Accuracy of PerFreezeClip in different local epochs of calendar elements when $\tau^*_{w_g} = 0.9$ and C = 2, where $\psi = 35$ and epochs are taken as 2, 10, 20, 30, and 40.

Based on the findings of the investigation into the impact of adjusting the number of local iterations on PerFreezeClip, it can be concluded that an increase in the number of local epochs generally leads to enhanced test accuracy. However, it is important to note that the optimal value of $\psi$ may vary depending on the number of local epochs, and a larger number of local epochs may not always result in a proportional increase in training accuracy. Moreover, the comparison between PerFreezeClip and FedRep with varying numbers of local epochs demonstrated that PerFreezeClip outperformed FedRep in specific epochs, even with a smaller number of local epochs. Nevertheless, as the number of local epochs extended to 30 and 40, the training accuracy did not align with the anticipated outcomes, suggesting that the maximum value of norms equal to 30 was inadequate for these extended epochs. From the experiments, we conclude that the test accuracy of PerFreezeClip is jointly determined by the freezing ratio parameter, the maximum number of norms, and the number of local training rounds.

## 6. Conclusions

In this study, PerFreezeClip, a personalized federated learning (FL) algorithm, effectively addresses heterogeneous data challenges using adap_clip for gradient regulation and a freezing method for personalized parameter control. It excels in test accuracy compared to some other personalization methods while maintaining a lightweight design. Future work will explore layer adaptive learning rates to enhance gradient adjustments across different layers. In future research, we believe that we can explore the layer adaptive learning rate that dynamically adjusts the gradient across different layers based on the existing dynamic thresholding methods.

**Author Contributions:** Conceptualization, J.Z. and Z.L.; methodology, J.Z. and Z.L.; software, Z.L.; validation, J.Z. and Z.L.; formal analysis, J.Z. and Z.L.; investigation, J.Z. and Z.L.; resources, J.Z. and Z.L.; data curation, J.Z. and Z.L.; writing—original draft preparation, J.Z. and Z.L.; writing—review and editing, J.Z. and Z.L.; visualization, J.Z. and Z.L.; supervision, J.Z.; project administration, J.Z. and Z.L.; funding acquisition, J.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This paper was supported by the project "Research on Machine Learning Methods Based on Multi-party Participation" (20210101483JC), which is financially supported by the Science & Technology Development Program of Jilin Province, China.

**Data Availability Statement:** The datasets that support the results of this study are publicly available datasets, and the use of these datasets in this work adheres to the licenses of these datasets. The CIFAR10 dataset is available at http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz (accessed on 20 October 2023). The CIFAR100 dataset is now available at http://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz (accessed on 20 October 2023).

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Konečný, J.; McMahan, B.; Ramage, D. Federated optimization: Distributed optimization beyond the datacenter. *arXiv* **2015**, arXiv:1511.03575.
2. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
3. Dong, C.; Zhou, J.; An, Q.; Jiang, F.; Chen, S.; Pan, L.; Liu, X. Optimizing performance in federated person re-identification through benchmark evaluation for blockchain-integrated smart uav delivery systems. *Drones* **2023**, *7*, 413. [CrossRef]
4. Li, Z.; Sharma, V.; Mohanty, S.P. Preserving data privacy via federated learning: Challenges and solutions. *IEEE Consum. Electron. Mag.* **2020**, *9*, 8–16. [CrossRef]
5. Zhang, X.; Hong, M.; Dhople, S.; Yin, W.; Liu, Y. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. *arXiv* **2020**, arXiv:2005.11418.
6. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. *Proc. Mach. Learn. Syst.* **2020**, *2*, 429–450.
7. Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; Poor, H.V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7611–7623.
8. Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; Suresh, A.T. Scaffold: Stochastic controlled averaging for federated learning. In Proceedings of the International Conference on Machine Learning PMLR, Virtual, 13–18 July 2020; pp. 5132–5143.
9. Acar, D.A.E.; Zhao, Y.; Navarro, R.M.; Mattina, M.; Whatmough, P.N.; Saligrama, V. Federated learning based on dynamic regularization. *arXiv* **2021**, arXiv:2111.04263.
10. Li, Q.; He, B.; Song, D. Model-contrastive federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 10713–10722.
11. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [CrossRef] [PubMed]
12. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
13. Mansour, Y.; Mohri, M.; Ro, J.; Suresh, A.T. Three approaches for personalization with applications to federated learning. *arXiv* **2020**, arXiv:2002.10619.
14. Arivazhagan, M.G.; Aggarwal, V.; Singh, A.K.; Choudhary, S. Federated learning with personalization layers. *arXiv* **2019**, arXiv:1912.00818.
15. Bui, D.; Malik, K.; Goetz, J.; Liu, H.; Moon, S.; Kumar, A.; Shin, K.G. Federated user representation learning. *arXiv* **2019**, arXiv:1909.12535.
16. Liang, P.P.; Liu, T.; Ziyin, L.; Allen, N.B.; Auerbach, R.P.; Brent, D.; Salakhutdinov, R.; Morency, L.P. Think locally, act globally: Federated learning with local and global representations. *arXiv* **2020**, arXiv:2001.01523.
17. Smith, V.; Chiang, C.K.; Sanjabi, M.; Talwalkar, A.S. Federated multi-task learning. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
18. Corinzia, L.; Beuret, A.; Buhmann, J.M. Variational federated multi-task learning. *arXiv* **2019**, arXiv:1906.06268.
19. Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; Zhang, Y. Personalized cross-silo federated learning on non-iid data. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 7865–7873.

20. Collins, L.; Hassani, H.; Mokhtari, A.; Shakkottai, S. Exploiting shared representations for personalized federated learning. In Proceedings of the International Conference on Machine Learning PMLR, Online, 18–24 July 2021; pp. 2089–2099.
21. Babakniya, S.; Kundu, S.; Prakash, S.; Niu, Y.; Avestimehr, S. Federated sparse training: Lottery aware model compression for resource constrained edge. In Proceedings of the Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022), New Orleans, LA, USA, 2 December 2022.
22. Bibikar, S.; Vikalo, H.; Wang, Z.; Chen, X. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 6–10 November 2022; Volume 36.
23. Sidahmed, H.; Xu, Z.; Garg, A.; Cao, Y.; Chen, M. Efficient and private federated learning with partially trainable networks. *arXiv* **2021**, arXiv:2110.03450.
24. Pfeiffer, K.; Rapp, M.; Khalili, R.; Henkel, J. CocoFL: Communication-and computation-aware federated learning via partial NN freezing and quantization. *arXiv* **2022**, arXiv:2203.05468.
25. Seetharaman, P.; Wichern, G.; Pardo, B.; Le Roux, J. Autoclip: Adaptive gradient clipping for source separation networks. In Proceedings of the 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), Espoo, Finland, 21–24 September 2020; pp. 1–6.
26. Reddi, S.J.; Hefny, A.; Sra, S.; Poczos, B.; Smola, A. Stochastic variance reduction for nonconvex optimization. In Proceedings of the International Conference on Machine Learning PMLR, New York, NY, USA, 20–22 June 2016; pp. 314–323.
27. Zaheer, M.; Reddi, S.; Sachan, D.; Kale, S.; Kumar, S. Adaptive methods for nonconvex optimization. In Proceedings of the Advances in Neural Information Processing Systems 31 (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018.