AdaDP-CFL: Cluster Federated Learning with Adaptive Clipping Threshold Differential Privacy

Tao Yang and Xuebin Ma*
Inner Mongolia University, Hohhot, China
32109271@mail.imu.edu.cn, csmaxuebin@imu.edu.cn

***Abstract*—Federated learning is a distributed machine learning approach that enables multiple clients to train models collaboratively. As its data remains stored locally on each client, this approach significantly enhances the protection of private information. However, federated learning still faces privacy leakage risks in environments with data heterogeneity. Differential privacy mechanisms are widely utilized in federated learning to ensure privacy for clients, and the magnitude of the clipping threshold directly impacts model utility. The current research does not adequately address the impact of model accuracy and training loss on clipping thresholds and is challenged by excessive hyperparameter adjustments. In response to these challenges, we propose an adaptive clipping-based differential privacy federated learning algorithm named AdaDP-CFL. It achieves model personalization and facilitates knowledge sharing among different groups through clustering and regularization techniques. Subsequently, the algorithm addresses the issue of adaptive clipping for various clients, formulated as a Markov decision process, by utilizing a deep deterministic policy gradient model based on gradient differences across client groups. Experimental results demonstrate that our algorithm outperforms current algorithms in accuracy, effectively balancing privacy protection and model utility.**

***Index Terms*—personalized federated learning, data heterogeneity, sample-level differential privacy, adaptive clipping**

I. Introduction

Federated learning (FL) [1] is a machine learning paradigm that enables multiple clients to collaboratively train a model without sharing data. In this iterative process, clients transmit only the locally updated model parameters to a server for aggregation, thus constructing a global model. However, in scenarios where client data is non-independent and identically distributed (non-IID), utilizing a shared global model may lead to slow convergence or client drift [2], impacting the overall accuracy of the model.

Personalized federated learning is designed to tackle the challenge of data heterogeneity, allowing clients to train local models customized to their unique data distributions. Several methods have been proposed for implementing personalized federated learning: Parameter Decoupling [3] enables clients to share the base layer while retaining the personalized layer locally; Regularization [4] facilitates personalization by minimizing the divergence between local and global models; Clustering [5] categorizes clients into groups based on similar data distributions, optimizing a group model for each cluster. In clustering-based federated learning algorithms, data of clients in different groups may overlap or share similar characteristics. However, current research predominantly concentrates on optimization strategies within individual groups, neglecting the significance of knowledge sharing between groups, which impacts the accuracy of local models.

Moreover, privacy leakage remains a concern in personalized federated learning. Although local data is not directly shared in personalized federated learning, attackers may be able to infer data privacy from transmitted model parameters or gradients through inference attacks [6] or model inversion attacks [7]. Recent studies have explored the use of homomorphic encryption [8], secure multi-party computation [9], and differential privacy [10] to address privacy leakage in federated learning. However, homomorphic encryption and secure multi-party computation have high computational costs despite their robust security. In contrast, differential privacy balances privacy protection with lower computational overhead.

Numerous studies have focused on using differential privacy mechanisms to address privacy leakage in federated learning. In federated learning employing sample-level differential privacy [11] to protect data privacy, a clipping threshold is implemented to limit sensitivity and Gaussian noise is added to aggregated gradients. However, significant gradient variances can arise in non-IID scenarios. [12]. Applying a fixed clipping threshold in all clients may lead to disproportionate information loss, hindering model convergence and reducing accuracy. Recent research has explored adaptive clipping threshold, but their effectiveness largely depends on the choice of historical gradients and hyperparameters, such as the number of historical rounds and percentiles [13]-[14]. Finding an appropriate clipping threshold for clients in non-IID settings remains challenging, as improper settings might overlook important historical data and trends, and unsuitable percentile settings can lead to inaccurate threshold determination.

We propose a new framework, AdaDP-CFL, that first uses singular value decomposition and hierarchical clustering to group clients with similar data distributions. Then, the framework transforms the client clipping threshold selection problem into a Markov decision process and uses a deep deterministic policy gradient model to find optimal clipping threshold for each client aligning with gradient heterogeneity. Finally, a nearest neighbor optimization strategy is employed to facilitate inter-group model learning, aiding clients in optimizing local models on the server side. The main contributions of this paper are summarized as follows:

*Corresponding author.

- We propose the AdaDP-CFL, an adaptive clipping clustered federated learning framework. This framework groups clients with similar data distributions by applying singular value decomposition and hierarchical clustering, for each cluster independently constructs a model and trains the model for each group. We perform gradient clipping at the sample level and then add noise during differential privacy training for clients. Furthermore, a nearest neighbor optimization strategy is employed on the server side to enhance the learning process between different group models.
- We introduce an adaptive clipping threshold selection algorithm based on deep deterministic policy gradient (DDPG). The challenge of selecting a clipping threshold is formulated as a Markov decision process, where the DDPG model is strategically applied to determine the optimal thresholds for each client, thus achieving a balance between privacy protection and model performance.
- While evaluating multiple datasets, we found that the proposed AdaDP-CFL algorithm demonstrated higher model accuracy than existing algorithms while ensuring privacy protection.

The paper is organized as follows: Section II reviews related work on personalized federated learning and differential privacy within federated learning; Section III introduces the fundamentals of federated learning and differential privacy; Section IV elaborates on the proposed AdaDP-CFL framework; Section V presents the experimental results; and Section VI concludes the paper.

II. Related work

A. Personalized Clustered Federated Learning

To address the challenges of non-IID data in federated learning, personalized federated learning has been developed to offer personalized solutions. Clustered federated learning, as a subset of personalized federated learning groups clients with similar data distributions facilitating group-based federated learning. Ghosh et al. [5] proposed the IFCA algorithm for iteratively clustering clients based on model parameter similarity, with clients in the same group sharing base layers and only aggregating personalized layers. Sattler et al. [15] introduced the CFL algorithm, a federated multi-task learning framework that leverages the geometric properties of federated learning loss to dynamically group clients. Duan et al. [16] proposed the FlexCFL algorithm, which groups clients based on optimization direction similarity and provides an efficient mechanism for integrating new clients during training. Vahidian et al. [17] proposed the PACFL algorithm for iteratively clustering clients based on analyzing the principal angles between client data subspaces to identify distributional similarities among clients effectively, and there still exists a risk of privacy leakage. Xiao et al. [18] proposed the CFMTL algorithm, which, although it leverages the concept of multi-tasking still poses privacy risks by indirectly measuring the similarity of local data among clients through client model parameters, requiring the number of clusters to be predetermined.

These methods in personalized federated learning demonstrate enhancements in handling non-IID data. However, conventional clustered federated learning primarily concentrates on intra-group optimization, often overlooking the potential for inter-group knowledge sharing. Sharing knowledge across different group models could potentially enhance the accuracy of each model [19]. These approaches have limitations regarding privacy protection, not adequately addressing the privacy leakage associated with the transmission of parameters in personalized federated learning.

B. Differential Privacy in Federated Learning

Differential privacy ensures user privacy in federated learning processes. Current research on differential privacy in federated learning is divided into user-level and sample-level. Geyer et al. [20] protected clients' datasets by adding noise to model parameters on the server side, thus ensuring the undisclosed participation of any client. Wei et al. [11] proposed the UDP algorithm assuming a server is not trusted, setting thresholds for clipping, and adding noise to the model parameters of each client before uploading to the server. Andrew et al. [21] introduced an algorithm for adaptive clipping threshold for model parameters, which adaptively adjusts thresholds to track specific percentiles of update norms and adds Gaussian noise. Noble et al. [22] limited sensitivity with per-sample gradient clipping and added Gaussian noise before uploading to the server. Xu et al. [23] proposed the ADADP algorithm employing a strategy for adaptive clipping and noise addition like RMSPROP, based on the mean squared variance of historical gradients. Wang et al. [13] introduced the DP-FedMeta algorithm in the federated meta-learning setting, using a sliding window to store historical gradient information and selecting clipping thresholds through percentiles.

Adaptive clipping algorithms in differential privacy federated learning primarily rely on historical gradient information to predict the clipping threshold of the next round. Current methods may not fully capture the impact of accuracy and training loss on gradients, potentially leading to the selection of a suboptimal clipping threshold. Moreover, determining an appropriate number of historical gradient rounds and computing the optimal clipping threshold from these historical gradients entails considerable computational effort.

III. Basic technology

A. Federated learning

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

$$w = \sum_{i=1}^{N} p_i w_i \tag{3.1}$$

where w_i are the model parameters trained by the i-th client, w are the model parameters aggregated by the server, N is

TABLE I: List of Main Symbols

Symbols	Describe
T	Rounds
i	Client index
K	Number of clusters
N	Number of clients
s	Cluster index
G_s	Cluster model parameters
G_s^{t+1}	The s-th cluster's cluster model weight in round $t+1$
D_i	Dataset for client i
A	Similarity relationships between different sets of models
L	The effect of penalty term
$G_s^{t+1,old}$	Group model before optimization
$G_s^{t+1,new}$	Group model after optimization
η	Local learning rate
γ	Cluster threshold
C	clipping threshold
C_i^{t+1}	The i-th client's clipping threshold in round $t+1$
$w_{s,i}^{t+1}$	The s-th cluster's i-th client model weight in round $t+1$

the number of clients, $p_i = \frac{|D_i|}{|D|} \geq 0$ and $\sum_{i=1}^{N} p_i = 1$, $|D| = \sum_{i=1}^{N} |D_i|$ is the total number of all data samples. This optimization problem can be expressed as:

$$w^* = \arg\min_{w} \sum_{i=1}^{N} p_i F_i(w, D_i) \tag{3.2}$$

In the equation above, $F_i(w, D_i)$ is the local loss function of the i-th client. In federated learning, N clients collaborate under the server to train the model jointly, ultimately optimizing (3.2) to converge to the global optimum. To address this optimization task, federated learning employs multiple rounds of communication between the local training and server aggregation phases, as shown in Fig. 1. In local training, clients initialize and train the global model locally, and then upload the updates. The server aggregates these models to update and redistribute the global model.

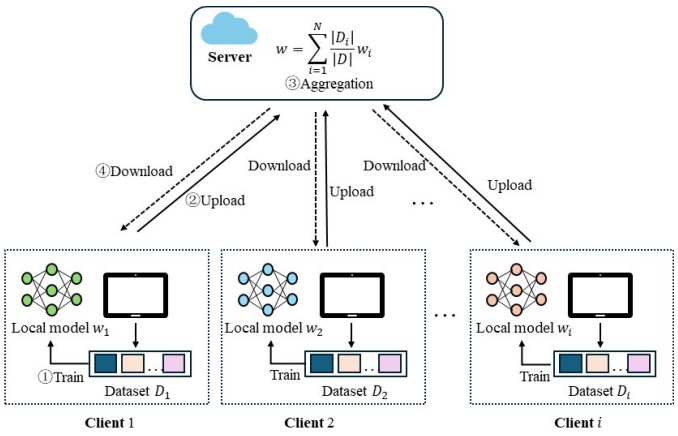


Fig. 1: Federated Learning

B. Differential privacy

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

Definition 1 (Differential Privacy [10]): Let $M : D \to R$ be a random mechanism, D and D' are neighboring datasets that differ by at most one record. If for any output $S \subseteq R$ the random mechanism M satisfies the following equation on both D and D', then the mechanism M satisfies (ε, δ)-DP:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S] + \delta \tag{3.3}$$

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

Definition 2 (Rényi Divergence[24]): Given two probability distributions P and Q, the Rényi-α divergence between P and Q is defined as:

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left[\left(\frac{P(x)}{Q(x)} \right)^{\alpha} \right] \tag{3.4}$$

Definition 3 (Rényi Differential Privacy [24]): For any two adjacent datasets D and D', if there exists a random mechanism $M : D \to R$ that satisfies the following equation, then the random mechanism M is said to satisfy (α, ε)-RDP:

$$D_\alpha(M(D) \parallel M(D')) \leq \varepsilon \tag{3.5}$$

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

C. Truncated singular value decomposition

Truncated singular value decomposition [25] (TSVD), derived from singular value decomposition, is a widely used method for matrix decomposition. SVD decomposes a matrix into the product of three matrices: $A = U\Sigma V^T$. Here, $U = [u_1, u_2, \ldots, u_p]$ and $V = [v_1, v_2, \ldots, v_p]$ are orthogonal matrices, while Σ is a diagonal matrix with its diagonal elements known as singular values.

$$A = \begin{bmatrix} u_1 & \ldots & u_p \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_p \end{bmatrix} \begin{bmatrix} v_1 & \ldots & v_p \end{bmatrix} \tag{3.6}$$

Truncated singular value decomposition is a dimensionality reduction technique that simplifies a matrix by retaining only its largest singular values and discarding the smaller ones, thereby efficiently.

IV. METHOD

In this section, we first present the schematic of the AdaDP-CFL framework, as shown in Fig. 2. And the main symbols used in the paper, as listed in Table 1. Subsequently, we detail the algorithms related to the training process. Finally, we demonstrate that the proposed algorithms satisfy differential privacy guarantees.

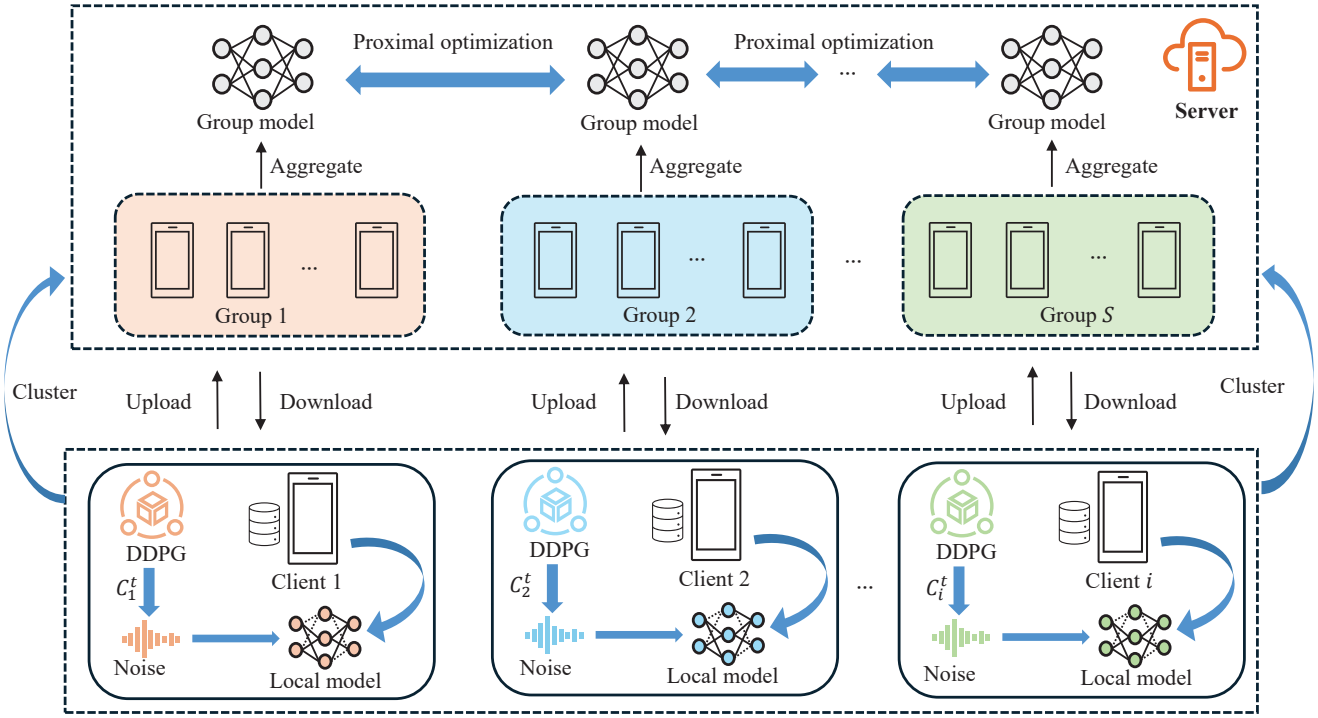


Fig. 2: AdaDP-CFL framework

A. *Cluster Federated learning*

In the initial stages of federated learning, truncated singular value decomposition is employed by each client to process local data, which then informs the clustering operations conducted on the server. Each client performs singular value decomposition on its data matrix D_i, selects the leading p left singular values as the grouping vector, and subsequently adds Gaussian noise to the partitioned vector to obtain $\tilde{U}^p$.

Each client transmits its vector of left singular values to the server. The server calculates the smallest principal angles between the uploaded vectors using (4.1) to construct the similarity matrix A upon receiving them. In this matrix, a smaller value of A_{ij} indicates greater similarity between datasets i and j:

$$A_{ij} = \theta_1(\tilde{U}_i^p, \tilde{U}_j^p), \quad i, j = 1, \ldots, N \tag{4.1}$$

Subsequently, the server employs hierarchical clustering on the similarity matrix A to group the clients. This approach allows the server to form a suitable number of groups automatically without predetermining the number of clusters.

During the training phase, model parameters w^0 are initially initialized. The server randomly selects available clients and broadcasts w^0 to them. The clients commence training on their local data and execute differential privacy stochastic gradient descent to update the model, thereby obtaining the updated model parameters:

$$w_i^t = w_i^{t-1} - \eta\tilde{g} \tag{4.2}$$

The perturbed gradient $\tilde{g}$ is obtained through adaptive clipping and the addition of noise, details of which are elaborated in the following section. For clients within the same cluster, an aggregated group model G_s^t is formed through intra-cluster aggregation.

$$G_s^t = \sum_{i \in s} \frac{|D_i|}{|D|} w_i^t \tag{4.3}$$

The server independently optimizes each group model G_s^t by solving the following optimization problem, facilitating inter-group learning, and obtaining the updated group model:

$$G_s^{t,new} = G_s^t - a\nabla \mathrm{Prox}(G) \tag{4.4}$$

$$\mathrm{Prox}(G) = \frac{1}{2\eta}\|G_s^t - G_s^{t-1}\|_F^2 + L\sum_{j=1}^{K} \lambda_{s,j}\|G_s^t - G_j^t\|^2 \tag{4.5}$$

where $\|\cdot\|_F$ represents the Frobenius norm used to measure the size of a matrix, L represents the degree of regularization, and $\lambda_{s,j}$ represents the similarity between group models s and j. $\mathrm{Prox}(G)$ is the regularization term, which ensures consistency between the group model G_i^t at iteration t and its previous iteration G_i^{t-1} while also promoting knowledge sharing with models from other groups, thereby facilitating knowledge transfer among models and improving overall learning performance.

$$\lambda_{s,j} = \sum_{j=1}^{K} \cos\{G_s^t, G_j^t\} \tag{4.6}$$

After receiving all updated models in each round, the server computes the normalized cosine similarity between various group models using (4.6). This measurement is employed to ascertain the degree of similarity across the different group models.

Algorithm 1 Cluster Federated Learning with Adaptive clipping threshold Differential Privacy

Input: rounds T, cluster threshold γ, number of clients N, learning rate η
Output: group model G

1: Initialize global model parameters w^0
2: **for** each round $t = 1, 2, \ldots, T$ **do**
3: Select a subset of clients randomly
4: **//Client Update**
5: **for** each client $i \in N_t$ in parallel **do**
6: **if** $t = 1$ **then**
7: client i sends $\tilde{U}_p$ to the server
8: $A \leftarrow$ server forms A based on $\theta_1(\tilde{U}_p^i, \tilde{U}_p^j)$
9: server groups clients $(G_1, \ldots, G_K)$ by HC (A, γ)
10: initializing all clusters with w^0
11: **else**
12: **for** $e = 1, \ldots, E$ **do**
13: **for** batch $b \in B$ **do**
14: $g^t(x_j) \leftarrow \nabla L(W_i, x_j)$
15: $g^t(x_j) \leftarrow g^t(x_j) / \max(1, \frac{\|g^t(x_j)\|_2}{C})$
16: $\tilde{g}^t \leftarrow \frac{1}{L} \sum_{i=1}^{L} g^t(x_j) + N(0, \sigma^2 C^2)$
17: $w_i \leftarrow w_i - \eta \tilde{g}^t$
18: **end for**
19: **end for**
20: **end if**
21: $(C_1^t, C_2^t, \ldots, C_K^t) \leftarrow$ adaptive clipping threshold selection
22: sends group ID and weights to the server
23: **//Server Update**
24: **for** each cluster $s \in K$ **do**
25: $G_s^t \leftarrow$ server aggregates client models with the same ID
26: $\lambda \leftarrow$ server compute group parameters similarity by (4.6)
27: $G_s^{t,new} \leftarrow$ server update by proximal-optimization by (4.4)
28: client i receives the corresponding cluster model from the server
29: **end for**
30: **end for**
31: **end for**

In Algorithm 1, we initially initialize the global model parameters w^0 (line 1). Each round randomly samples N clients for federated learning (lines 2-3). All clients extract and upload their left singular value vectors to the server before training begins (lines 6-7). The server constructs the similarity matrix A based on the minimum principal angle (line 8), groups clients using hierarchical clustering (line 9), and initializes all group models (line 10). During training, clients train on their local datasets and apply differential privacy mechanisms to perturb gradients obtaining $\tilde{g}^t$, thus updating local models w_i (lines 12-19). Clients use Algorithm 2 for adaptive threshold clipping selection for the next round (line 21). The noise-added model parameters and corresponding group IDs are sent to the server by the clients (line 22). The server performs group aggregation calculates the similarity between different group models, and conducts proximal optimization upon receiving these parameters before sending optimized group models to respective clients (lines 25-28).

B. Adaptive Clipping

1) Problem Definition: In the process of per-sample gradient clipping, when the L_2 norm of a training sample's gradient surpasses the threshold C, each gradient component of that sample is scaled down in proportion to its gradient L_2 norm. This clipping limits gradient sensitivity within a finite range, and Gaussian noise is then added to the aggregated gradients for differential privacy protection. If the clipping threshold C is too small, the clipped gradient may point in a direction opposite to the true gradient, affecting model convergence. If C is too large, more noise is added to the gradient due to the variance of Gaussian noise being proportional to $\sigma \times C$.

During the t-th round of training, we define two metrics to quantify the contribution of the clipping threshold to the performance of the local model:

Definition 4 (loss difference): The difference in local model training loss values between w_i^{t-1} and w_i^t is denoted as ϕ_1, defined as follows:

$$\phi_1 = \text{loss}_i^{t-1} - \text{loss}_i^t \tag{4.7}$$

where loss_i^{t-1} and loss_i^t represent the test loss values for the local model at client i on dataset D_i during the $t-1$-th and t-th rounds of training, respectively.

Definition 5 (accuracy difference): The difference in local model training accuracy between w_i^{t-1} and w_i^t is denoted as ϕ_2, defined as follows:

$$\phi_2 = \text{acc}_i^t - \text{acc}_i^{t-1} \tag{4.8}$$

where acc_i^{t-1} and acc_i^t represent the test accuracy for local model at client i on dataset D_i during the $t-1$-th and t-th rounds of training, respectively.

The local contribution can be quantified by a linear combination of (4.6) and (4.7), as demonstrated below:

$$\beta_1 \cdot \phi_1 + \beta_2 \cdot \phi_2 \tag{4.9}$$

where $\beta_1, \beta_2 \in (0, +\infty)$. The goal of the t-th round of local training can be simply stated as maximizing the local contribution by choosing an appropriate C_i^t, defined as:

$$\max_{C_i} \{\beta_1 \cdot \phi_1 + \beta_2 \cdot \phi_2\} \tag{4.10}$$

Therefore, the clipping threshold parameter selection in the entire local training process can be expressed as the following optimization problem, denoted as P_0, defined as:

$$P_0 : \max_{C_i} \sum_{t=1}^{T} \{\beta_1 \cdot \phi_1 + \beta_2 \cdot \phi_2\}$$

where $C_i = (C_i^1, C_i^2, \ldots, C_i^T)$ is the sequence of clipping thresholds selected across different rounds.

2) Problem Transformation: In a personalized federated learning environment, clients encounter diverse data distributions necessitating dynamic adaptation of clipping threshold to foster model convergence towards the optimum. Since P_0 is a static optimization problem, it is transformed into a Markov decision process P_1 for dynamic threshold selection. This involves defining a triplet to describe the process of selecting the clipping threshold for clients.

State Space: The difference in loss and accuracy at the local client is updated at the end of each communication round, reflecting the federated learning process. The state space is represented as:

$$s_i^t = \{\text{loss}_i^t, \text{acc}_i^t\} \tag{4.11}$$

Action Space: An action for the current round is selected and executed as defined upon receiving the state from a previous round:

$$a_i^t = \{c_i^t\} \tag{4.12}$$

Reward: In state s_i^t, the local model contribution obtained by executing action a_i^t is considered the current reward, defined as:

$$r_i^t = R(s_i^t, a_i^t, s_i^{t+1}) = \max_{C_i} \{\beta_1 \cdot \phi_1 + \beta_2 \cdot \phi_2\} \tag{4.13}$$

With this triplet, the P_0 problem is converted into the MDP P_1:

$$P_1 : \max_{\mu_i} \sum_{t=1}^{T} \{\gamma^{t-1} \cdot r_i^t\}$$

where $\gamma \in (0, 1)$ is the discount factor for rewards and $\mu_i : s_i^t \to a_i^t$ is the action generation function.

3) Problem Resolution: To address the Markov Decision Process P_1 given the continuous action space for the clipping threshold, we adopt the deep deterministic policy gradient model and design a corresponding Actor-Critic architecture [26]. In this framework, the actor-network generates actions based on the current state, while the critic network evaluates the expected return of a particular action. Specifically, the actor-network takes the current states $\{\text{loss}_i^t, \text{acc}_i^t\}$ as input and outputs actions a_i^t within the clipping threshold C_i^t. In contrast, the critic network assesses the value of this action given state s_i^t and action a_i^t. Through iterative training, the DDPG model continually updates the parameters of both the actor and critic networks, minimizing the prediction error of the critic network and guiding the actor-network to generate more effective action strategies.

Algorithm 2 DDPG-Assisted Adaptive Clip-Threshold Selection

Input: state space dimension $s_i^t = \{\text{loss}_i^t, \text{acc}_i^t\}$, action-space dimension $a_i^t = \{C_i^t\}$, learning rates $\{\text{lr}_\mu, \text{lr}_Q\}$, soft-update rate β
Output: action space dimension $a_i^{t+1} = \{C_i^{t+1}\}$

1: Initial actor-network μ with weights θ_μ, and critic-network Q with weights θ_Q
2: Initialize target networks μ' and Q' with weights $\theta_{\mu'} \leftarrow \theta_\mu$, $\theta_{Q'} \leftarrow \theta_Q$
3: Initialize replay buffer B
4: **for** $t = 1, \ldots, T$ **do**
5: Select a subset of clients randomly
6: **for** each selected client i **do**
7: **if** $t \leq 2$ **then**
8: Initialize randomly action a_i^t
9: **else**
10: Compute reward r_i^t based on loss_i^t and acc_i^t
11: Store transition $(s_i^{t-1}, a_i^{t-1}, r_i^t, s_i^t)$ in B
12: Sample a minibatch from B
13: Update critic-network by minimizing the mean squared error loss across all sampled tuples
14: Update critic-network by the gradient descent method:
15: $\theta_i^{Q,t} \leftarrow \theta_i^{Q,t} - \text{lr}_Q \cdot \nabla_{\theta_i^Q} \text{loss}_i(t-1)$
16: Update actor-network using the sampled policy gradient:
17: $\theta_i^{\mu,t} \leftarrow \theta_i^{\mu,t} + \text{lr}_\mu \cdot \nabla_{\theta_i^\mu} J_i(t-1)$
18: Soft updates the target networks with a mix of target and main network weights:
19: $\theta_{\mu'} \leftarrow \beta\theta_\mu + (1-\beta)\theta_{\mu'}$
20: $\theta_{Q'} \leftarrow \beta\theta_Q + (1-\beta)\theta_{Q'}$
21: **end if**
22: **end for**
23: Determine new action a_i^{t+1} from actor-network
24: Apply action a_t to client's clip-threshold for DDPG-SGD
25: **end for**

Initially, the actor and critic networks along with their target network copies are initialized, and an experience replay buffer is established for subsequent learning (lines 1-3). In each training round, a group of clients is randomly selected, and the following steps are iterated for each client (line 5): if it is the first two rounds, a random initial action is chosen. (lines 7-8); otherwise, the reward is calculated based on the training loss and accuracy of the previous state (lines 9-10). The state, action, reward, and next state are stored in the experience replay buffer (line 11). A mini-batch of experiences is sampled from the buffer (line 12), and the critic network is updated by minimizing the mean squared error across all samples (line 13). The parameters of the critic network are updated using gradient descent, and the actor-network is updated using policy gradients (lines 14-17). A soft update strategy is then

employed to gradually adjust the parameters of the target network towards those of the leading network (lines 18-20). Actions for the next step are chosen based on the policy generated by the actor-network. Finally, the clipping threshold determined by the actor-network is applied in the differential privacy stochastic gradient descent for clients (lines 23-24).

C. Privacy Analysis

In this section, we analyze the privacy guarantees of the proposed algorithm.

Theorem 1 (Gaussian Mechanism of RDP[24]). Let $S = \max_{D,D'} \|M(D) - M(D')\|_2$ be the L_2 sensitivity of function M. Then, for the Gaussian mechanism: $M(D) + N(0, \sigma^2 I)$, it satisfies $(\alpha, \frac{\alpha S^2}{2\sigma^2})$-RDP.

Theorem 2 (Conversion from RDP to DP[24]). If M is a (α, ϵ)-RDP mechanism, then for $\forall \delta \in (0, 1)$, mechanism M also satisfies (ϵ', δ)-DP, where ϵ' is defined as follows:

$$\epsilon' = \epsilon + \frac{\log(1/\delta)}{\alpha - 1} \tag{4.14}$$

Theorem 3 (Sequential composition theorem of RDP[24]). Let $M_1 : \mathcal{D} \to \mathcal{R}_1$ satisfy (α, ϵ_1)-RDP; $M_2 : \mathcal{D} \to \mathcal{R}_2$ satisfy (α, ϵ_2)-RDP. Then the sequential composition mechanism $M_{1,2}$ of M_1 and M_2 satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.

Theorem 4 (Post-processing[10]). Suppose a mechanism satisfies (ϵ', δ)-DP and another randomized mechanism A. The mechanism M after the operation of mechanism A still satisfies (ϵ, δ)-DP.

Proof. Each client incorporates Gaussian noise satisfying differential privacy into their local model training. It suffices only to prove that the privacy budget for the i-th client is (ϵ', δ). Let g and g' be the private gradients for two adjacent datasets D and D', and for a given sensitivity S we have:

$$S = \max_{D,D'} |g - g'| \tag{4.15}$$

According to the Gaussian mechanism of Rényi Differential Privacy (Theorem 1) for all $\forall \delta \in (0, 1)$, we have:

$$D_\alpha(N(0, \sigma^2 I) \| N(S, \sigma^2)) = \frac{\alpha S^2}{2\sigma^2} \tag{4.16}$$

From the equation above, it is evident that the uploading process of the clients satisfies $(\alpha, \alpha S^2/2\sigma^2)$-Rényi Differential Privacy, or (α, ϵ)-RDP. Combining this with the post-processing mechanism (Theorem 4), when Algorithm 1 processes the output of the differential privacy mechanism in multiple local models after server aggregation, it still satisfies differential privacy. Then, the RDP is sequentially combined, aggregating the privacy loss over multiple rounds (Theorem 3). Finally, the obtained $(\alpha, \alpha S^2/2\sigma^2)$-RDP is converted to (ϵ, δ)-DP using Theorem 2.

V. Evaluation

A. Models and Datasets

Datasets: we employ the CIFAR10, FMNIST, SVHN, and CIFAR100 datasets for training, as shown in Table II.

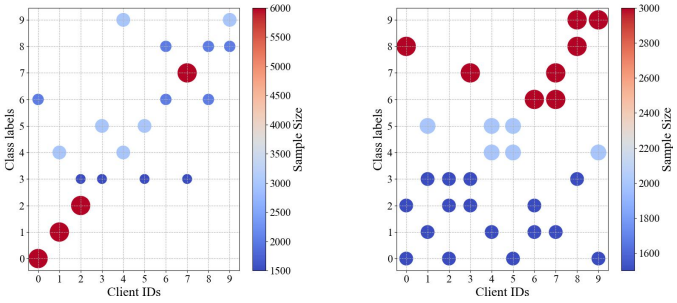
TABLE II: Statistics of the Dataset

Dataset	Samples	Class	Task
FMNIST	70000	10	Image classification
SVHN	73257	10	Image classification
CIFAR10	60000	10	Image classification
CIFAR100	60000	100	Image classification

Models: we employ a Convolutional Neural Network for training on the CIFAR10, FMNIST, and SVHN datasets, and a ResNet9 network for the CIFAR100 dataset. The CNN has 2 conv layers with 6 and 16 output channels and 5×5 kernels, plus 3 FC layers with 120, 84, and 10 outputs. ResNet9 has 1 initial conv layer and 3 groups, each with a head layer and a residual block.

B. Implementation Details

Heterogeneity Level: We initially randomly allocate $p\%$ of the total available labels in the dataset to each client, followed by a random distribution of samples for each label to the clients possessing that label, as shown in Fig.3.



(a) $p\%$=20% (b) $p\%$=30%

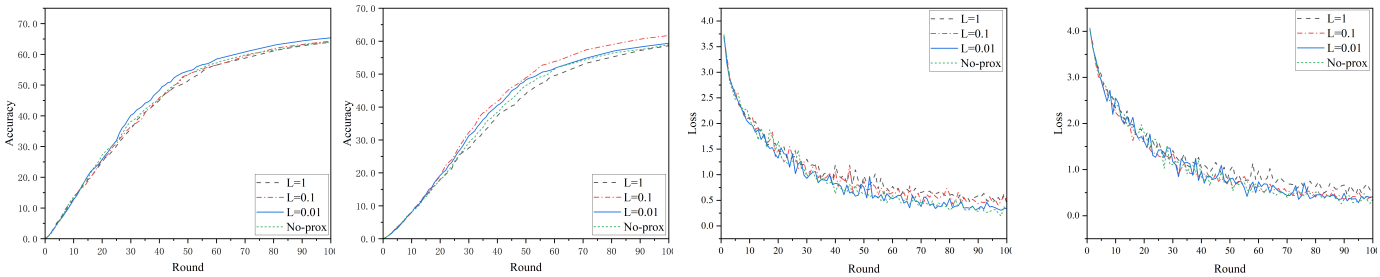
Fig. 3: Heterogeneity level

Comparison Algorithms: we utilize Fedavg[1], SCAFFOLD[4], IFCA[5], CFL[15], DP-Fedavg[27], DP-SCAFFOLD[27], DP-AGR[13] as benchmark algorithms.

Hyperparameter Settings: we set a global round number of 100, local iteration rounds to 5, batch size to 20, learning rate at 0.01, client count at 100, sampling rates at {0.1, 0.5}, momentum magnitude at 0.5 and regularization coefficients at {1, 0.1, 0.001}.

C. Results and Analysis

Tables III and IV evaluated the average accuracy for five FL algorithms across four datasets under two distinct data distribution scenarios with 20% and 30% label skew. The algorithms are Fedavg, Scaffold, IFCA, CFL, and one we proposed in this paper. For the proposed algorithm, different regularization parameters L=1.0, 0.1, and 0.01 are set to achieve optimal performance. The findings from both tables indicate that except for slight underperformance in the FMNIST dataset at a 30% label skew compared to the No-prox approach, the proposed algorithm consistently attains higher accuracy than the other evaluated FL algorithms. Fig.4 visually



(a) 20% non-IID label skew (b) 30% non-IID label skew (c) 20% non-IID label skew (d) 30% non-IID label skew

Fig. 4: Test accuracy and loss of ResNet9 on CIFAR100 with varied regularization parameters

compares accuracy and loss between the No-prox approach and algorithms with varying degrees of regularization under label imbalance conditions. At 20% label imbalance, the algorithm configured with L=0.01 outperforms the No-prox approach by approximately 1.5%. At 30% label imbalance, the algorithm configured with L=0.01 achieves an accuracy improvement of approximately 3% over the No-prox approach. In both scenarios of label skew, all curves demonstrate a trend of decreasing test loss with an increasing number of training rounds. In the graphs, the absence of Prox regularization leads to a faster decrease in test loss, but the loss curves exhibit fluctuations in the later stages of training. At 20% label skew, the setting of L=0.01 achieves a relatively lower test loss after 100 training rounds, whereas L=1 results in the highest loss. At 30% label skew, the L=0.1 setting performs best in the later phases. The regularization coefficient L is crucial in balancing personalization and knowledge sharing. The optimal L value varies depending on the data distribution and dataset, as demonstrated in Tables III and IV. At 20% label skew, a smaller L is chosen to reduce mutual learning. While at 30% label skew, a larger L is preferred to enhance learning from other groups. The overlap in sample labels and quantities across different clients grows with increased label skew. Thus, a larger L aids in learning shared knowledge more effectively. Conversely, with decreased label skew and fewer overlapping samples, a smaller L is selected to minimize interference from other group models, enhancing the effectiveness of personalized models.

TABLE III: Comparison of accuracy of different algorithms with 20% label skew

Algorithm	CIFAR10	FMNIST	SVHN	CIFAR100
FedAvg	46.570	85.250	84.531	41.560
Scaffold	85.812	97.254	94.523	61.240
IFCA	86.112	97.580	95.335	62.900
CFL	60.922	81.225	72.639	40.290
No-Prox	86.844	97.777	95.471	63.971
Ours(L=1)	86.542	97.536	94.967	63.971
Ours(L=0.1)	**87.190**	97.701	95.174	64.334
Ours(L=0.01)	87.188	**97.822**	**95.670**	**65.409**

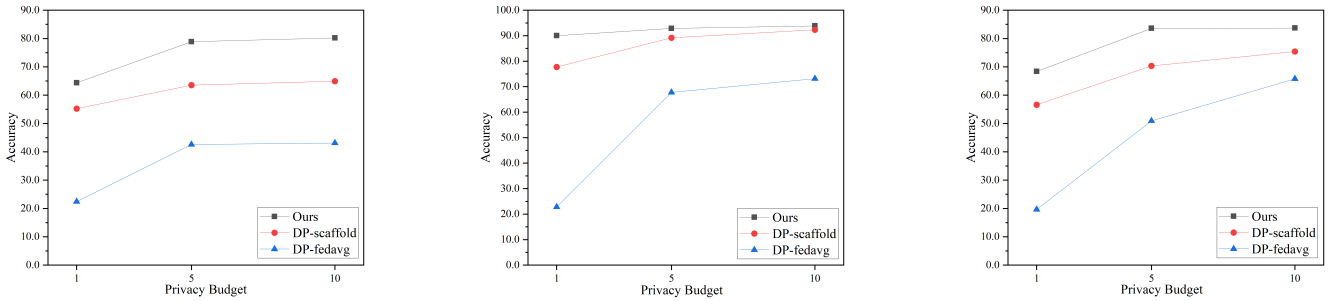
This section presents the test accuracy of DP-FedAvg, DP-Scaffold, and our algorithm under various privacy budgets (ϵ = 1, 5, 10) across datasets experiencing a 20% label skew. As depicted in Fig. 5, the accuracy of all three methods

TABLE IV: Comparison of accuracy of different algorithms with 30% label skew

Algorithm	CIFAR10	FMNIST	SVHN	CIFAR100
FedAvg	49.020	87.150	86.374	44.870
Scaffold	77.062	95.563	91.109	58.933
IFCA	77.188	95.739	92.060	56.507
CFL	57.465	88.129	64.725	37.230
No-Prox	76.805	**96.236**	92.280	58.948
Ours(L=1)	**77.932**	95.624	91.381	58.628
Ours(L=0.1)	76.385	95.850	92.398	**61.734**
Ours(L=0.01)	75.917	95.760	**92.456**	59.393

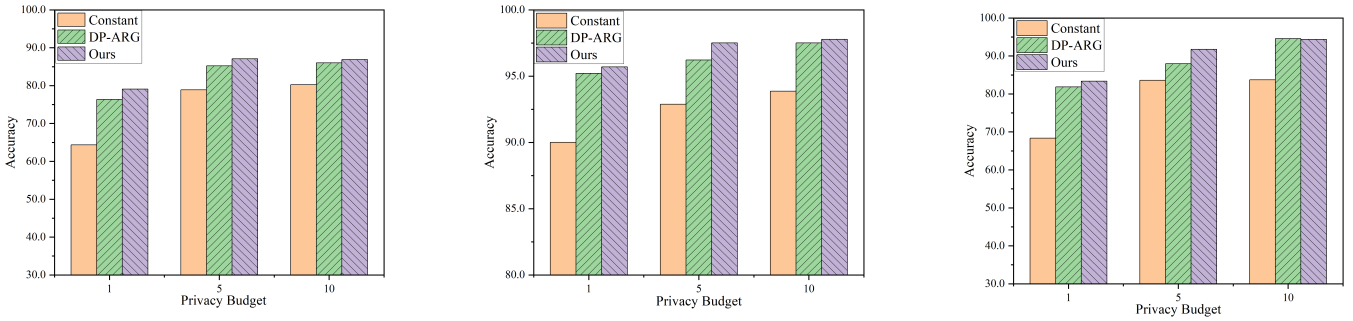
decreases as the noise level increases, yet our algorithm's accuracy consistently surpasses that of the other two baselines. It is observed that as the privacy budget increases, there is a corresponding rise in model accuracy. Specifically, compared to DP-FedAvg, our algorithm exhibits significant performance improvements. DP-FedAvg is prone to model bias due to the influence of data distribution heterogeneity, often resulting in lower accuracy. However, our algorithm also shows a marked increase in accuracy compared to DP-Scaffold. While DP-Scaffold effectively addresses data heterogeneity and enhances privacy protection, its performance decreases in federated learning environments with only partial client participation. In contrast, our proposed algorithm maintains high accuracy despite limited client involvement. Additionally, the architecture of our algorithm employs clustering and regularization techniques, effectively mitigating challenges caused by data heterogeneity. Particularly, the regularization techniques reduce the negative impacts of noise by minimizing excessive model weight and disturbances, thereby significantly enhancing the model's robustness to noise.

In this section, we experimentally evaluate the proposed adaptive clipping threshold algorithm and compare it with a fixed clipping threshold approach and the DP-AGR adaptive clipping algorithm. Specifically, the DP-AGR adopts a clipping strategy that involves retaining a history of gradient norms and using the 90th percentile of these past five rounds' norms as the clipping threshold for subsequent rounds. Results in Fig.6 show that our adaptive clipping algorithm outperforms the fixed threshold approach across various differential privacy budgets (ϵ = 1, 5, 10). The adaptive clipping algorithm significantly enhances model accuracy compared to the fixed



(a) CIFAR10 Dataset (b) FMNIST Dataset (c) SVHN Dataset

Fig. 5: Test accuracy for different ϵ in CNN models on various datasets



(a) CIFAR10 Dataset (b) FMNIST Dataset (c) SVHN Dataset

Fig. 6: Test accuracy using adaptive clipping threshold for different ϵ in CNN models on various datasets

clipping threshold method. Since the fixed clipping threshold cannot automatically adjust based on changes in the gradient norm, this approach often leads to considerable model bias and loss of information. When the privacy budget is low and the model is in the later stages of training, the sensitivity to noise increases, and excessive noise addition at this stage can significantly reduce model accuracy. Conversely, when the privacy budget is higher, retaining more model information becomes crucial for enhancing accuracy because less noise is added. Our algorithm demonstrates superior accuracy in all settings except for SVHN at a privacy budget of 10 compared to DP-AGR. In an evaluation using the CIFAR10 dataset, our algorithm outperforms DP-AGR by 2.73% in accuracy when the privacy budget is set to 1 and by 1.85% when the privacy budget is set to 5. This discrepancy primarily arises from the additional noise introduced by DP-AGR during processing, which impacts the gradient norms. Particularly, using historical percentiles as clipping threshold under low privacy budgets fails to capture the true magnitude of gradient norms accurately. Another key factor in the performance decline of DP-AGR is the setting of parameters for the storage period of historical gradient norms and the selection of percentile, which vary across different settings.

VI. Conclusion

We introduce the AdaDP-CFL algorithm to enhance privacy protection in personalized federated learning under data heterogeneity. The clipping threshold selection on the client side is formulated as a Markov decision process, utilizing a deep deterministic policy gradient model to adjust the clipping threshold for each client during training dynamically. We employ proximal optimization techniques to enhance knowledge sharing between model groups on the server side, thereby enhancing model performance. We apply the AdaDP-CFL on deep learning networks using real datasets and demonstrate that AdaDP-CFL outperforms previous methods.

Acknowledgment

This paper is supported by the Inner Mongolia Natural Science Foundation (Grant No. 2023MS06022), the University Youth Science and Technology Talent Development Project (Innovation Group Development Plan) of Inner Mongolia A. R. of China (Grant No. NMGIRT2318), the "Inner Mongolia Science and Technology Achievement Transfer and Transformation Demonstration Zone, University Collaborative Innovation Base, and University Entrepreneurship Training Base" Construction Project (Supercomputing Power Project) (Grant No. 21300-231510), and the Self-project Program of Engineering Research Center of Ecological Big Data, Ministry of Education.

References

[1] J. Konecnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies

for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, vol. 8, 2016.

[2] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[3] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[4] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.

[5] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 586–19 597, 2020.

[6] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[7] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[8] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.

[9] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, "Secure multi-party computation: theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2019.

[10] C. Dwork, "Differential privacy," in *International colloquium on automata, languages, and programming*. Springer, 2006, pp. 1–12.

[11] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE transactions on information forensics and security*, vol. 15, pp. 3454–3469, 2020.

[12] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[13] N. Wang, Y. Xiao, Y. Chen, N. Zhang, W. Lou, and Y. T. Hou, "Squeezing more utility via adaptive clipping on differentially private gradients in federated meta-learning," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 647–657.

[14] J. Fu, Z. Chen, and X. Han, "Adap dp-fl: Differentially private federated learning with adaptive noise," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 656–663.

[15] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[16] M. Duan, D. Liu, X. Ji, Y. Wu, L. Liang, X. Chen, Y. Tan, and A. Ren, "Flexible clustered federated learning for client-level data distribution shift," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2661–2674, 2021.

[17] S. Vahidian, M. Morafah, W. Wang, V. Kungurtsev, C. Chen, M. Shah, and B. Lin, "Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, 2023, pp. 10 043–10 052.

[18] Y. Xiao, J. Shu, X. Jia, and H. Huang, "Clustered federated multi-task learning with non-iid data," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2021, pp. 50–57.

[19] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Sharing knowledge in multi-task deep reinforcement learning," *arXiv preprint arXiv:2401.09561*, 2024.

[20] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[21] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy, "Differentially private learning with adaptive clipping," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 455–17 466, 2021.

[22] M. Noble, A. Bellet, and A. Dieuleveut, "Differentially private federated learning on heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 10 110–10 145.

[23] Z. Xu, S. Shi, A. X. Liu, J. Zhao, and L. Chen, "An adaptive and fast convergent approach to differentially private deep learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1867–1876.

[24] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 2017, pp. 263–275.

[25] P. C. Hansen, "Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 3, pp. 503–518, 1990.

[26] Q. Chen, Z. Wang, J. Chen, H. Yan, and X. Lin, "Dap-fl: Federated learning flourishes by adaptive tuning and secure aggregation," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[27] M. Noble, A. Bellet, and A. Dieuleveut, "Differentially private federated learning on heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 10 110–10 145.