Academic Journal of Applied Sciences
ISSN: 3105-0549 (Print), ISSN: 3105-0557 (Online) | Vol. 1, No. 1, 2026

Personalized Hierarchical Federated Learning Algorithm Based on Adaptive Differential Privacy

Huijuan Jia [1], Jingyu Zhao [2, *]

[1] College of Software, Henan Polytechnic University, Henan454000, China
[2] College of Computer Science and Technology, Henan Polytechnic University, Henan 454000, China
*** Corresponding author**

Abstract: This paper proposes ADP-PHFL, an Adaptive Differential Privacy-based Personalized Hierarchical Federated Learning framework. It integrates hierarchical federated learning with personalized training via knowledge transfer, mitigating performance degradation caused by directly applying a global model while reducing communication overhead. An adaptive differential privacy mechanism is introduced to allocate privacy budgets more effectively, avoiding the drawbacks of uniform noise injection and enhancing privacy protection against data leakage. Theoretical analysis establishes the convergence and privacy guarantees of the proposed method. Experimental results demonstrate that ADP-PHFL achieves improved communication efficiency and higher model accuracy.

Keywords: Adaptive Differential Privacy; Personalized Federated Learning; Hierarchical Federated Learning.

1. Introduction

Recent advances in deep learning have fundamentally transformed numerous application domains, including image processing, natural language processing, and video analysis. To date, deep learning models are predominantly trained on powerful computational platforms, such as cloud data centers, which host centrally aggregated large-scale datasets. However, in many practical scenarios, data are generated and distributed across edge devices (e.g., smartphones and sensors), and transferring such data to centralized servers for model training introduces increasingly severe privacy concerns. Consequently, privacy-preserving distributed training has attracted significant attention. In 2017, Google proposed Federated Learning (FL) and the Federated Averaging (FedAvg) algorithm, enabling the training of deep learning models without centralizing data. In this paradigm, local devices download a global model from the cloud server, perform multiple rounds of local training, and upload updated model parameters for aggregation. This process is repeated until the desired model accuracy is achieved.

Federated learning achieves distributed training by decomposing the process into two stages: parallel local model updates on clients and global model aggregation on the server. The most common architecture adopts a client–server paradigm. However, this approach has notable limitations. On the one hand, as the number of clients increases, communication can become a bottleneck, resulting in slower model updates and increased latency. Wu Q et al.[1] proposed a personalized scheme to address Non-IID distributions in health monitoring data; however, the large data volume significantly increases communication overhead during model aggregation. On the other hand, curious or untrusted servers may pose threats to user privacy. To address these challenges, hierarchical federated learning (HFL) architectures have been proposed[2]. In HFL, intermediate layers can perform partial computations, thereby reducing the burden on the central server. Moreover, intermediate nodes are typically less privacy-sensitive than centralized servers, which helps enhance user privacy protection. Chen et al.[3] proposed a random sampling-based HFL method that selects a subset of clients in each training round, effectively reducing communication costs per round. Nevertheless, this sampling strategy may lead to insufficient gradient representativeness and does not incorporate privacy protection mechanisms or address model adaptation under data heterogeneity.

Furthermore, federated learning faces significant challenges due to data heterogeneity. When local datasets exhibit Non-IID characteristics, employing a shared global model may lead to suboptimal performance. Personalization has emerged as an effective strategy to address this issue, aiming to adapt the global model to individual participants to improve their performance. Representative approaches include transfer learning (Transfer Learning, TL) proposed by Pan S. J. et al.[4] and data augmentation methods by Maharana K. et al. [5]. Transfer learning can be integrated into the federated learning framework to construct personalized models by transferring knowledge (i.e., pretrained model parameters) from a source domain to related target domains. Federated Transfer Learning (FTL) [6] extends this idea by enabling participants to personalize the global model, thereby alleviating statistical heterogeneity. Accordingly, this chapter focuses on hierarchical personalized federated learning.

Differential privacy (DP) has been widely applied in federated learning to provide rigorous privacy guarantees by injecting carefully calibrated random noise during the client-side model upload stage. Its core mechanism relies on a privacy budget to quantitatively control the strength of privacy protection. Shi L. et al. [7] incorporated differential privacy into a hierarchical federated learning framework, achieving a balance between privacy and performance; however, their method adopts a globally uniform and fixed privacy budget allocation strategy, without considering differences in data volume and gradient characteristics across clients. Li S. et al. [8] combined personalized training with differential privacy in HFL, enabling client-specific privacy budget allocation and partially balancing privacy protection and model utility. Nevertheless, their approach considers limited allocation dimensions and does not incorporate

dynamic factors such as training rounds and gradient magnitudes for adaptive adjustment, leaving room for improvement in precision. To address these limitations, this chapter introduces adaptive differential privacy, which assigns distinct privacy budgets based on user-specific characteristics to better balance model utility and privacy protection.

To overcome the issues of uneven privacy budget allocation, suboptimal model aggregation, and high communication overhead leading to increased latency, this paper proposes a novel solution with the following contributions:

(1) An Adaptive Differential Privacy–Personalized Hierarchical Federated Learning framework, termed ADP-PHFL, is proposed. It integrates personalized federated learning into a hierarchical architecture and leverages knowledge transfer to enable client-specific model training, thereby mitigating performance degradation caused by directly applying a global model. Meanwhile, the hierarchical structure alleviates the communication overhead introduced by knowledge transfer.

(2) An adaptive differential privacy mechanism is incorporated to allocate privacy budgets more effectively. This approach avoids the inefficiency and performance degradation caused by uniformly adding noise to all parameters, while further enhancing privacy protection and preventing potential user data leakage within the hierarchical framework.

(3) Theoretical analysis is provided to establish the convergence and privacy guarantees of the proposed method. Extensive experimental results demonstrate its superior performance in terms of communication efficiency and model accuracy.

2. Preliminary

(1) Hierarchical Federated Learning

Hierarchical Federated Learning (HFL) is an improved distributed learning framework proposed to address the limitations of traditional federated learning, including high communication overhead, slow aggregation speed, and potential privacy leakage. Unlike the conventional client–server architecture, HFL introduces a three-tier structure consisting of clients, edge nodes, and a central server. By partitioning the data transmission process and adopting hierarchical aggregation and scheduling strategies, HFL enables more efficient, stable, and privacy-preserving model training.

In HFL, edge nodes play a critical role as intermediate aggregators and coordinators. They first perform local aggregation on model updates uploaded by nearby or regionally clustered clients, and then transmit the aggregated regional models to the central server. As a result, the server only needs to process information from a limited number of edge nodes, rather than directly aggregating updates from a large number of clients, thereby significantly reducing communication overhead. Moreover, by preventing raw client data or direct updates from being transmitted to the central server, HFL effectively mitigates the risk of privacy leakage.

(2) It has the characteristics of freehand brushwork

Federated Transfer Learning (FTL) integrates the principles of federated learning and transfer learning, aiming to enhance model generalization and personalization in heterogeneous data environments while preserving data privacy. This approach is particularly suitable for real-world scenarios with Non-IID data distributions, such as cross-regional healthcare modeling, inter-enterprise collaborative risk control, and cross-device intelligent recommendation systems.

Transfer learning emphasizes leveraging existing models or prior knowledge to facilitate new tasks, thereby mitigating the challenges caused by limited data availability or distribution discrepancies through feature sharing or parameter transfer. Incorporating transfer learning into the federated learning framework gives rise to federated transfer learning, which has become a promising solution to the aforementioned challenges.

The core idea of FTL is to transfer the learned global model, or part of its parameters, to participating clients after collaborative federated training, followed by further local adaptation and optimization to achieve personalized modeling. During the federated stage, participants collaboratively train a global model through multiple rounds of communication, enabling it to capture shared feature representations across clients. Subsequently, in the transfer stage, each participant uses the global model as an initialization or feature extractor and performs fine-tuning or incremental learning on local data, thereby obtaining a personalized model that better aligns with its own data distribution. This paradigm effectively reduces the risk of negative transfer caused by distribution heterogeneity while maintaining privacy constraints.

From a technical perspective, FTL typically involves two key mechanisms. The first is the construction of shared feature representations, where federated training is used to learn generalizable lower-layer representations or a common subnetwork. The second is the design of personalized adaptation layers, where client-specific components—such as dedicated layers, partially trainable parameters, or task-specific modules—are introduced to accommodate individual data distributions. In addition, some approaches further enhance cross-client knowledge transfer through techniques such as knowledge distillation, multi-task learning, and parameter decomposition. Compared with traditional federated learning methods that rely primarily on simple parameter aggregation, FTL emphasizes hierarchical model design and knowledge-sharing mechanisms, thereby improving overall system stability and training efficiency.

3. System Model

This section presents the system architecture and workflow of ADP-PHFL, which are divided into two stages: global learning and client-side personalization, as illustrated in the figure 1.

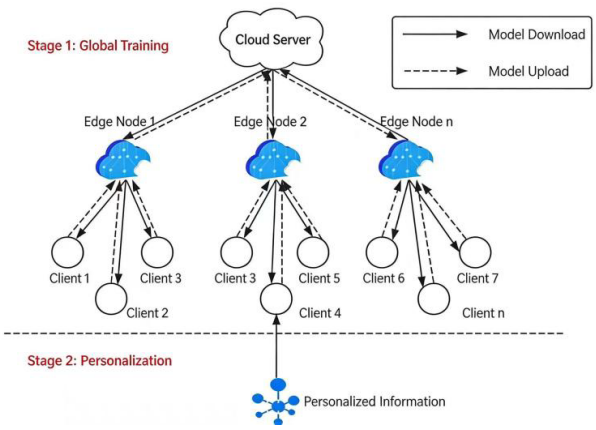

Figure 1. ADP-PHFL System Model

(1) Global Learning Stage:

First, the first-tier server broadcasts the global model weights to all edge nodes, and the second-tier edge nodes further distribute the model to the active clients in the third tier. Each client then performs local training and applies adaptive noise injection to the updated model parameters to satisfy differential privacy requirements. The noise-perturbed model updates are transmitted to the corresponding edge nodes, where local aggregation is conducted. The aggregated regional models are subsequently sent to the central server. The server then aggregates the received models to update the global model and broadcasts the updated model back to all edge nodes. This process is iteratively repeated until convergence is achieved.

(2) Client Personalization Stage:

The edge nodes broadcast the global model received from the server to all clients. Each client utilizes its local dataset to adapt the global base model, which encapsulates transferred general knowledge. Specifically, the lower-layer feature extraction layers are frozen to preserve shared representations, while the upper layers are fine-tuned using local data to adjust personalized parameters, thereby generating client-specific personalized models.

4. Methodology

The proposed ADP-PHFL algorithm introduces an adaptive differential privacy mechanism into a hierarchical federated learning framework, enabling personalized training while ensuring user data privacy. Its objective is to enhance privacy protection and communication efficiency in Non-IID environments. The algorithm adopts a three-tier collaborative architecture consisting of a cloud server, edge servers, and clients. The cloud server is responsible for maintaining and updating the global model, edge servers perform local aggregation, and clients conduct personalized training and privacy-preserving operations based on local data.

At the beginning of training, the cloud server initializes the global model parameters and distributes them to all edge servers, enabling local training to proceed from a unified model. Subsequently, each edge server randomly selects a subset of clients according to a predefined sampling ratio to participate in the current training round, thereby reducing communication overhead and improving scalability. The selected clients use the current edge model as initialization and perform several steps of gradient-based optimization on their local datasets to obtain updated model parameters.

To prevent the inference of raw client data from model updates, an adaptive differential privacy mechanism is incorporated at the client side. After local training, the gradient vectors are clipped using the L2 norm to bound the sensitivity of individual updates within a predefined threshold. An adaptive privacy budget allocation function is then constructed based on the client's data size, the current training round, and the norm of the update vector. Specifically, clients with larger datasets are assigned relatively larger privacy budgets to reduce the impact of noise on model accuracy; a round-dependent decay factor progressively decreases the privacy budget over training rounds to control overall privacy consumption and strengthen privacy protection in later stages; and a gradient-based adjustment factor dynamically regulates the budget to suppress the impact of anomalous updates on system stability. After determining the privacy budget, Gaussian noise is injected into the clipped updates to generate differentially private model updates.

At the edge layer, each server aggregates the privacy-preserving client updates using a data-size-weighted averaging strategy to obtain updated edge model parameters. This strategy mitigates bias caused by data imbalance and reduces the impact of noise on the overall model. After several rounds of local aggregation at the edge level, the updated models are uploaded to the cloud server, where a further weighted aggregation based on the total data volume of each edge node is performed to produce a new global model. This process is iterated until convergence.

After global training is completed, a personalization phase is introduced to further improve model adaptability to local data distributions. Each client uses the final edge model as initialization and performs a small number of fine-tuning steps on its local dataset to obtain a personalized model. This stage does not participate in global aggregation, thereby better preserving local feature information.

Overall, the ADP-PHFL algorithm effectively integrates adaptive differential privacy with personalized training within a hierarchical federated learning framework. On the one hand, gradient clipping and Gaussian noise injection provide rigorous privacy guarantees for client data. On the other hand, dynamic privacy budget allocation and hierarchical weighted aggregation mitigate the negative impact of noise on model accuracy while ensuring privacy protection. Furthermore, the personalization stage enhances performance under heterogeneous data distributions, achieving a balanced trade-off among privacy preservation, system stability, and model accuracy. The complete pseudocode of ADP-PHFL is presented in Algorithm 1.

Algorithm 1: ADP-PHFL

Input:Dataset D_i^j of client i under edge node j ; set of all edge nodes E;set of clients C^j associated with edge node j ; local batch size B ; early stopping patience Patience; learning rate η.

Output: Final personalized model parameters w_i^j for each client.

1:Initialize the global model $w(0)$ at the cloud server.

2:**for** each training round $t=0,1,\ldots,N_{rounds}-1$ **do**

3: **for** each edge node $j\epsilon E$ **do**

4: Distribute the global model to all edge nodes:$w^j \leftarrow w(t)$

5: end for

6: **for** $t_1=1,2,...,N_{agg}$ **do**

7: **for** each edge node $j\epsilon E$ **do**

8: Compute the number of participating clients: $k \leftarrow \max(|C^j|.F,1)$

9: Randomly select k clients C_k^j

10: **for** each client $i \in C_k^j$ **do**

11: Initialize client model:$w_i^j \leftarrow w^j$

12: Perform local training:$w_i^j \leftarrow \text{ClientLocalTraining}(i,w_i^j)$

13: end for

14: **for** each client $i \in C_k^j$ **do**

15: Gradient clipping:$g_i^t \leftarrow \frac{g_i^t}{\max(1,/\!/ g_i^t /\!/_2/C)}$

16: Adaptive privacy budget:$e_i^t=\epsilon_0 \cdot \frac{|D_i^j|}{D_{max}^j} \cdot \frac{1}{\sqrt{t+1}} \cdot \frac{1}{1+\alpha /\!/ g_i^t /\!/_2}$

17: Noise scale:$\sigma_i^t=\frac{C\sqrt{2\ln(1.25/\delta)}}{\epsilon_i^t}$

18: Sample Gaussian noise:$n_i^t \sim N(0,(\sigma_i^t)^2 I)$
19: Noisy update:$w_i^j \leftarrow w^j + g_i^t + n_i^t$
20: **end for**
21: Clients upload models to the corresponding edge node
22: Edge aggregation: $w^j \leftarrow$ EdgeAggregation $\left(\{w_i^j\}_{i \in c^j}\right)$
23: end for
24: end for
25: Edge nodes upload models to the cloud server
26: Global aggregation: $w(t) \leftarrow$ CloudAggregation $\left(\{w^j\}_{j \in E}\right)$
27:end for
28:**for** each edge node $j \epsilon E$ **do**
29: Distribute the final global model:$w^j \leftarrow w(t)$
30:end for
31:**or** each edge node $j \epsilon E$ **do**
32: **for** each client $i \in C^j$ **do**
33: Client receives edge model: $w_i^j \leftarrow w^j$
34: Client personalization:$w_i^i \leftarrow$ ClientPersonalization $\left(i, w_i^j\right)$
35: end for
36:end for
37:EdgeAggregation $\left(\{w_i^j\}_{i \in C^j}\right)_{i \in C^j}$
38: Weighted averaging by client data size:$w^j \leftarrow \sum_{i \in C_k^j} \frac{\lvert D_i^j \rvert}{\sum_{l \in C_k^j} \lvert D_l^j \rvert} \times w_i^j$
39: **return** w^j
40:CloudAggregation $\left(\{w^j\}_{j \in N}\right)$
41: Weighted averaging by edge data size:$w(t) \leftarrow \sum_{j \in E} \frac{\sum_{i \in C^j} \lvert D_i^j \rvert}{\sum_{u \in E} \sum_{l \in C^u} \lvert D_l^u \rvert} \times w^j$
42: return $w(t)$

5. Experiment

The hardware and software configurations used in this study are consistent with those described in Chapter 3. Two public image classification datasets, MNIST and CIFAR-10, are employed for evaluation. The MNIST dataset is designed for handwritten digit recognition (0–9) and contains 60,000 training samples and 10,000 test samples, all represented as 28×28 grayscale images. The CIFAR-10 dataset consists of 32×32 color images across 10 classes, with a total of 60,000 samples, including 40,000 training samples, 10,000 test samples, and 10,000 validation samples.

The configurations of the convolutional neural networks (CNNs) differ slightly for each dataset. For MNIST, the network consists of two convolutional layers, two pooling layers, one dropout layer, and two fully connected layers. For CIFAR-10, the network includes three convolutional layers, one pooling layer, and two fully connected layers.

The proposed ADP-PHFL method is applied to image classification tasks, where CNN models are trained on both MNIST and CIFAR-10 datasets. In the simulation, clients and edge servers at the same hierarchy level share identical parameter settings. Mini-batch stochastic gradient descent (SGD) is adopted for local model updates. Data distributions are configured under both IID and Non-IID settings.

To systematically evaluate the impact of statistical heterogeneity, this section controls data partitioning across clients using a Dirichlet distribution on the CIFAR-10 dataset. Three Non-IID scenarios with varying heterogeneity levels are constructed: Dir(100), Dir(5), and Dir(0.1), corresponding to low, moderate, and high heterogeneity, respectively. A larger Dirichlet parameter results in more uniform data distribution (closer to IID), while a smaller parameter leads to more skewed distributions and stronger heterogeneity.

Four baseline methods are considered: a hierarchical federated learning method without privacy protection, a random sampling-based hierarchical method, a hierarchical method with fixed differential privacy budget, and a personalized hierarchical method with differential privacy.

Under low heterogeneity, client data distributions are relatively balanced, leading to consistent gradient directions and minimal aggregation conflicts. All methods converge rapidly and achieve high accuracy. The method without privacy protection achieves the best performance, while the introduction of differential privacy slightly reduces accuracy due to noise perturbation. In contrast, the proposed ADP-PHFL achieves higher accuracy than fixed-budget methods and exhibits smoother convergence, demonstrating strong stability and robustness.

Under moderate heterogeneity, data imbalance becomes more pronounced, increasing gradient inconsistency. All methods experience slower convergence and reduced accuracy. Methods with fixed privacy budgets suffer more significant performance degradation due to amplified instability caused by noise. Although random sampling and personalized methods alleviate communication overhead, their adaptability to statistical heterogeneity remains limited. In comparison, ADP-PHFL maintains higher accuracy by combining hierarchical aggregation with personalized training, effectively preserving both shared and local feature representations.

Under high heterogeneity, most clients contain only a small number of classes, leading to highly inconsistent gradients. Methods with differential privacy exhibit slow and unstable convergence, while the non-private method maintains relatively higher accuracy but with noticeable oscillations. The proposed ADP-PHFL maintains a relatively stable convergence trend and achieves better robustness under extreme heterogeneity.

Overall, as the degree of heterogeneity increases, model accuracy decreases, convergence slows, and fluctuations become more pronounced. This indicates that statistical heterogeneity is a key factor limiting federated learning performance. The combination of differential privacy noise and gradient bias further exacerbates instability. The proposed method effectively mitigates these issues through hierarchical design, adaptive privacy budget allocation, and personalization, demonstrating strong robustness and generalization capability.

In hierarchical federated learning, the participation ratio and scale of clients and edge servers significantly affect convergence performance and final model accuracy.

From the perspective of client participation rate, increasing the participation ratio accelerates convergence and improves accuracy. At low participation rates, fewer clients participate in each round, leading to higher gradient variance and increased instability. When the participation rate reaches a sufficiently high level, model performance stabilizes,

indicating a favorable trade-off between communication efficiency and accuracy. Moreover, under differential privacy, low participation rates amplify the impact of noise, while higher participation helps mitigate this effect.

In contrast, the impact of edge server participation rate is relatively minor. Even when the participation ratio decreases, the final model accuracy remains largely unchanged, with only slight differences in convergence speed. This is because client updates are first aggregated at the edge level, making the system robust to partial edge node absence.

Regarding system scale, increasing the number of clients improves model performance due to richer data diversity and broader coverage. However, when the number becomes excessively large, the effective participation ratio per round decreases, and the accumulation of privacy noise leads to slower convergence and reduced accuracy.

The number of edge servers also plays an important role. A smaller number of edge nodes leads to more stable aggregation and higher accuracy. As the number increases, each edge node covers fewer clients, reducing the representativeness of local models and increasing gradient conflicts during global aggregation. Excessive edge nodes result in unstable convergence and performance degradation. Therefore, the number of edge nodes should be carefully configured according to system scale and data distribution.

To analyze the effect of aggregation frequency, two key parameters are considered: k1, which controls the aggregation frequency between clients and edge nodes, and k2, which controls the aggregation frequency between edge nodes and the cloud server.

Increasing k1 significantly improves model accuracy. When k1is too small, local training is insufficient, leading to unstable updates and excessive accumulation of differential privacy noise. As k1 increases, local models become more robust, resulting in more stable aggregation and improved accuracy. However, beyond a certain point, performance gains gradually saturate.

Conversely, increasing k2 leads to a decline in model accuracy. Frequent aggregation between edge nodes and the cloud server reduces the stability of local aggregation and increases the impact of noise due to repeated privacy composition.

In summary, a larger k1 and a smaller k2are beneficial for improving model performance. This configuration allows sufficient local optimization, reduces instability caused by frequent aggregation, and mitigates the negative impact of differential privacy noise.

6. Conclusion

This paper focuses on the challenges of privacy preservation and heterogeneous data adaptation in hierarchical federated learning, and proposes an Adaptive Differential Privacy-based Personalized Hierarchical Federated Learning framework (ADP-PHFL). Built upon a cloud–edge–client three-tier architecture, the proposed method integrates federated transfer learning with adaptive differential privacy. It reduces cross-node communication overhead through hierarchical aggregation, mitigates the impact of data heterogeneity via personalized training, and balances privacy protection and model performance through dynamic privacy budget allocation.

From a theoretical perspective, the privacy guarantees of the proposed method are formally established. From an experimental perspective, its effectiveness is validated by comparing model accuracy under varying degrees of data heterogeneity. Furthermore, its practical applicability is demonstrated in a cross-node image recognition scenario for intelligent security systems, where the method effectively adapts to heterogeneous monitoring nodes while satisfying data compliance requirements. Overall, the proposed approach provides an efficient and feasible solution for privacy-preserving learning under heterogeneous data conditions.

References

[1] Wu Q, Chen X, Zhou Z, et al. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring[J]. IEEE Transactions on Mobile Computing, 2020, 21(8): 2818-2832.

[2] L. Liu, J. Zhang, S. H. Song and K. B. Letaief, "Client-Edge-Cloud Hierarchical Federated Learning," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9148862.

[3] Chen, H. Jiang and Q. Hu, "Utility-Enhanced Personalized Privacy Preservation in Hierarchical Federated Learning," in IEEE Transactions on Mobile Computing, vol. 24, no. 6, pp. 5264-5279, June 2025, doi: 10.1109/TMC.2025.3531919.

[4] Pan S J, Yang Q. A survey on transfer learning[J]. IEEE Transactions on knowledge and data engineering, 2009, 22(10): 1345-1359.

[5] Maharana K, Mondal S, Nemade B. A review: Data pre-processing and data augmentation techniques[J]. Global Transitions Proceedings, 2022, 3(1): 91-99.

[6] He C, Annavaram M, Avestimehr S. Group knowledge transfer: Federated learning of large cnns at the edge[J]. Advances in neural information processing systems, 2020, 33: 14068-14080.

[7] Shi L, Shu J, Zhang W, et al. HFL-DP: Hierarchical federated learning with differential privacy[C]//2021 IEEE Global Communications Conference (GLOBECOM). IEEE, 2021: 1-7.

[8] Li S, Liu Y, Feng F, et al. HierFedPDP: Hierarchical federated learning with personalized differential privacy[J]. Journal of Information Security and Applications, 2024, 86: 103890.