Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(10): 7811–7832.
DOI: 10.3934/era.2026335
Received: 03 June 2026
Revised: 09 August 2026
Accepted: 20 August 2026
Published: 10 September 2026

Research article

Layer-wise FCFL: Federated learning with scaling-factor-based layer-wise fuzzy clustering

Milin Zhao[1,†], Guangwei Xu[1,†], Jianbo Lu[1,*], Haopu Lv[2], Wenxin Zhao[1], Peng Zhang[1], Liwei Li[1] and Yang Lu[3]

[1] School of Computer Science and Technology, Shandong University, Shandong Provincial Key Laboratory of Computing-Network Integration, Shandong University, Qingdao, China

[2] MCC Capital Engineering & Research Incorporation Limited, Beijing, China

[3] School of Robotics and Automation, Nanjing University, Suzhou, China

[†] These authors contributed equally to this work.

[*] **Correspondence:** Email: jblu@sdu.edu.cn.

Abstract: Federated learning is a distributed learning paradigm in which clients collaboratively train a shared model through local computation and server-side aggregation of model updates, without sharing raw data. However, under non-independent and identically distributed data settings, the aggregation performance of conventional FL often degrades significantly. Although clustered FL is an effective approach to mitigating data heterogeneity, existing methods still exhibit several limitations, including high computational overhead from full-model parameter clustering, insufficient characterization of layer-wise difference under monolithic whole-model clustering and aggregation, and inadequate information utilization caused by hard clustering. To address these issues, we proposed a new layer-wise fuzzy clustered FL paradigm that uses the scaling factors of batch normalization layers as local data representations. Low-dimensional BN scaling factors were used to replace full-model parameters for lightweight clustering. Then, we introduced a layer-wise independent clustering and aggregation mechanism that performs client clustering and model aggregation separately for different network layers to better capture layer-specific heterogeneity. Finally, fuzzy C-means (FCM) was applied at each layer to enable fuzzy clustering, allowing each client to belong to multiple clusters and enabling membership-weighted aggregation for improved information utilization. Extensive experiments on the MNIST, EMNIST, CIFAR-10, and CIFAR-100 datasets with SIMPLE-CNN and VGG16 demonstrated the effectiveness of the proposed approach over several state-of-the-art clustered FL baselines, with up to 7.10% gains in test accuracy over the strongest baseline. Further ablation studies showed that both the proposed fuzzy clustering mechanism and layer-wise aggregation strategy provide significant benefits, and their combination achieved the best overall performance.

Keywords: federated learning; data heterogeneity; fuzzy clustering; layer-wise clustering; batch normalization

1. Introduction

In the current wave of artificial intelligence (AI), large-scale and high-quality data are fundamental to training powerful models. However, as artificial intelligence becomes increasingly widespread across various industries, concerns about data privacy and security have grown, prompting many countries and regions to enact strict data protection regulations (e.g., the General Data Protection Regulation (GDPR)) [1]. These regulations raise significant barriers to cross-organization data-sharing. Holders of sensitive data are often reluctant to release it. The high cost of data integration further exacerbates data silos and fragmentation. Moreover, conventional data transfer can cause data owners to lose control over their data, diminishing its value and leaving the allocation of resulting benefits opaque. At the same time, the rapid growth of the Internet of Things and edge computing means that vast amounts of data are inherently distributed and cannot be readily centralized for processing. Without a compliant way to overcome data isolation, progress in AI will be substantially constrained [2, 3].

To address this problem, federated learning (FL) has emerged as a distributed collaborative learning paradigm in which raw data remain on local devices, while only local model updates are exchanged and aggregated in order to learn a shared model. As a pioneering work in FL, FedAvg puts this paradigm into practice by alternating multiple steps of local stochastic gradient descent (SGD) on clients with server-side weighted averaging of client models in each communication round [4]. This paradigm enables effective utilization of distributed data under privacy and compliance constraints, providing a feasible technical path toward breaking data silos and achieving secure and efficient collaborative learning.

In real-world applications, however, training data across different clients are usually not independent and identically distributed (non-IID). For example, in mobile keyboard next-word prediction, each user's typed text reflects personal language and usage habits, leading to markedly different word and context distributions across devices [5]. Data heterogeneity is a fundamental challenge in FL. It can directly slow down global training, reduce the final convergence accuracy, and lead to inferior overall modeling performance. Prior studies have shown that under highly skewed non-IID settings, the performance of convolutional neural networks trained with FedAvg can drop substantially, e.g., by up to 11% on MNIST, 51% on CIFAR-10, and 55% on keyword spotting (KWS) [6].

Effectively addressing data heterogeneity can significantly improve the generalization and robustness of the learned global model. This progress is crucial for unleashing the full potential of federated learning: enabling stable operation and high-performance global model training in realistic heterogeneous environments, and further promoting the transition of FL from academic research to industrial-scale deployments.

To tackle data heterogeneity, clustered federated learning has emerged as a widely adopted approach [7, 8]. It partitions clients into multiple groups (with relatively consistent intra-cluster data distributions) and trains a dedicated cluster model for each group, thereby alleviating the performance degradation caused by non-IID data. However, conventional hard clustered FL assigns each client to a single cluster, which limits both data utilization and classification accuracy. Although some recent studies have introduced soft clustering mechanisms, they still follow a monolithic clustering paradigm at the whole-model level and overlook the multi-layer architecture of modern deep neural networks. In practice, a neural network comprises multiple layers. Different layers learn features at varying levels of abstraction and thus should be treated differently.

To address these challenges, we propose a layer-wise fuzzy clustered federated learning method. For example, using Visual Geometry Group (VGG) blocks as basic units, we perform clustering and model aggregation separately for different parts of modern neural networks, thereby improving the final model accuracy. Moreover, existing methods typically represent clients using either whole model parameters or gradient-based updates. Although both can be informative, these representations are often high-dimensional, leading to substantial clustering costs. Instead, we leverage the scaling factors of batch normalization layers as the clustering representation, which can simultaneously maintain high clustering accuracy and improve computational efficiency.

The major contributions of this paper are fourfold:

- A lightweight data-distribution representation is introduced by using the scaling factors of batch normalization (BN) layers as local data representations, which reduces clustering computation while preserving clustering quality.
- Leveraging the multi-layer architecture of modern neural networks, we develop a layer-wise clustering-and-aggregation paradigm in which client clustering and model aggregation are performed independently for each layer.
- By integrating fuzzy clustering with the layer-wise design, the proposed federated learning method assigns soft memberships at each layer and performs membership-weighted fusion across multiple cluster models to better capture mixed client data distributions.
- Extensive experiments demonstrate that our approach improves accuracy over existing clustered FL methods. Moreover, ablation studies further verify the effectiveness of both the layer-wise design and the fuzzy clustering algorithm.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. In Section 3, we formulate the problem. Section 4 details the architecture and algorithm design of layer-wise FCFL (fuzzy clustering federated learning). Experimental evaluations of layer-wise FCFL are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. Related work

Federated learning. Federated learning (FL) enables collaborative model training without centralizing raw data by exchanging and aggregating local model updates, and thus has become a key technique for privacy-preserving distributed learning. A core challenge in FL is *data heterogeneity* (non-IID data), which often slows down convergence and significantly degrades the performance of standard aggregation methods such as FedAvg.

Handling data heterogeneity in federated learning. To address non-IID data, early studies mainly focused on improving optimization and aggregation under the assumption of learning a single global model shared by all clients. FedAvg [4] performs local SGD with periodic weighted averaging, but its performance can degrade significantly under strong heterogeneity [6]. FedProx [9] introduces a proximal regularization term to mitigate client drift and improve training stability in heterogeneous networks. To overcome the limitations of a single global model, current research has evolved along three typical directions: (i) *model personalization*, e.g., FedPer [10] that aggregates shared representation layers while keeping personalized heads, and Ditto [11] that learns a personalized model for each client with a proximal term; (ii) *knowledge transfer/distillation*, e.g., FedGEN [12] distills knowledge

on the server without relying on a public proxy dataset; and (iii) *clustered federated learning*, which mitigates heterogeneity by grouping clients based on data similarity and learning cluster-specific models. Representative methods include clustered federated learning (CFL) [7], iterative federated clustering algorithm (IFCA) [8], and privacy-preserving federated adaptation for effective model personalization (PFA) [13]. Since clustered FL is most relevant to our method, we next provide a more detailed review of clustered FL from both hard and soft clustering perspectives.

Hard clustered federated learning. Clustered federated learning (clustered FL) has been widely investigated for mitigating non-IID issues, where clients are partitioned into groups with relatively consistent intra-cluster distributions and each group trains a dedicated model. A representative early method is CFL [7], which recursively splits clients according to update similarity (e.g., cosine similarity) to form a hierarchy of clusters. IFCA [8] further formulates this process as an alternating optimization procedure, iteratively inferring client cluster identities via empirical risk minimization and updating cluster parameters. With a focus on privacy-preserving client grouping, PFA [13] uses sparse rectified linear unit (ReLU) activation representations for distance-based grouping, followed by FedAvg-based aggregation within each group. To address more dynamic environments, dynamic clustered federated learning (DCFL) [14] introduces a multi-stage adaptive clustering pipeline, while distribution-aware federated learning (DistFL) [15] extracts distribution knowledge from a pre-aggregated global model and then clusters uploaded client models based on the Kullback–Leibler (KL) divergence between their softmax output vectors.

Soft clustered federated learning. Despite the progress of hard clustering, hard assignment requires each client to belong to only one cluster, which can under-utilize information when data distributions overlap. Soft clustered FL alleviates this issue by allowing probabilistic memberships so that one client can contribute to multiple cluster models. For example, fuzzy clustering federated learning algorithm (FCFLA) [16] applies FCM-based memberships and aggregates updates with both membership weights and update directions. Along a similar line, federated learning with soft clustering (FLSC) [17] alternates between local training and multi-cluster assignment according to the cross-entropy losses of cluster models on local data. Federated learning with recursive fuzzy clustering (FedRFC) [18] extends this direction with recursive fuzzy clustering over reduced-dimensional gradient features, further improving adaptability across clients. Nevertheless, most existing soft clustered FL methods still perform clustering and aggregation at the *whole-model* level, overlooking the layered nature of modern deep neural networks.

Motivation for layer-wise fuzzy clustered FL. Modern networks are composed of multiple heterogeneous layers/modules, where different layers learn features at different abstraction levels. Therefore, clustering and aggregation at the whole-model level may be suboptimal. These observations motivate our layer-wise fuzzy clustered federated learning framework, which performs clustering and aggregation in a layer-wise manner (e.g., using VGG blocks) and adopts lightweight yet discriminative client representations.

3. Problem formulation

This section introduces the problem formulation of layer-wise FCFL. The important notation that will be commonly used later is summarized in Table 1.

We investigate a clustered federated learning method based on FCM clustering. Specifically, we

consider a system consisting of a cloud server and N clients. Each client i holds a local dataset $\mathcal{D}_i$. The local data distribution may differ significantly across clients (non-IID). To preserve data privacy, the server and clients communicate using a protocol in which only model parameters or updates are exchanged, while raw datasets are never shared. Our goal is to learn cluster-specific models (or block-wise cluster models W_{l,c_l} in the layer-wise setting) that better match different data distributions, where each client is represented by the batch normalization (BN) scaling factors $\gamma_{l,i}$ and participates in clusters via fuzzy memberships.

Table 1. Notation summary.

Symbol	Definition
N	Number of clients
i	Client index, $i \in \{1, \ldots, N\}$
$K,\ k$	Preset number of clustering centroids (clusters) in conventional clustered FL and cluster index, $k \in \{1, \ldots, K\}$
T_{kmeans}	Total iteration rounds of the K-means algorithm until convergence
T_{fcm}	Total iteration rounds of the FCM algorithm until convergence
$a_i,\ a$	Hard cluster assignment of client i and assignment vector $a = [a_1, \ldots, a_N]$
l	Network-block index, $l \in \{1, \ldots, L\}$
L	Number of network blocks in the model
$\mathcal{D}_i$	Local dataset of client i
W	Model parameters on the server
W_i	Model parameters of client i
W_k	Model parameters of cluster k (conventional clustered FL)
$W_{l,i}$	Model parameters of block l for client i
W_{l,c_l}	Model parameters for cluster c_l on block l
n_i	Local data size of client i
$f_d(\cdot),\ F_i(\cdot)$	Loss on the d-th local sample and local objective of client i
μ	Hyperparameter for the ℓ_1 regularization term
$\gamma_{l,i}$	BN scaling factor of block l for client i (feature vector for clustering)
d	Dimension of the feature vector for each sample
$c'_{l,i}$	Hard cluster index of client i at block l
C_l	Number of clusters for block l
$u_{l,c_l,i}$	Membership value of client i to cluster c_l at block l
U_l	Membership matrix at block l, $U_l = [u_{l,c_l,i}]$
m	The hyperparameter that controls how fuzzy the cluster will be
v_{l,c_l}	Cluster center vector of cluster c_l at block l
V_l	Center set at block l, $V_l = \{v_{l,c_l}\}_{c_l=1}^{C_l}$
$d(x, v)$	Cosine distance
$\mathbb{1}(\cdot)$	Indicator function (equals 1 if the condition is true, otherwise 0)
$\mu_B,\ \sigma_B^2,\ \epsilon$	Mini-batch mean, variance, and numerical-stability constant in BN
$\gamma,\ \beta$	BN scale and shift parameters in affine transformation
$x,\ \hat{x},\ y$	BN input activation, normalized activation, and affine output
$\lambda_{l,i}$	Lagrange multiplier for the membership-sum constraint of client i at block l

Conventional clustered FL typically aims to jointly learn client-to-cluster assignments and cluster-specific models. While minimizing local training losses, it groups clients with similar data distributions or model updates into the same cluster and maintains an independent global model for each cluster. To avoid notation overlap with our layer-wise formulation, let this conventional setting include N clients and K clusters, and denote the cluster assignment of client i by a_i. The local objective of client i for the full-model parameters W_i can be written as

$$F_i(W_i) = \frac{1}{n_i} \sum_{d=1}^{n_i} f_d(W_i). \tag{3.1}$$

Under hard clustering, where each client belongs to exactly one cluster, a representative joint optimization objective is

$$\min_{\{W_k\}_{k=1}^{K},\, a\in\{1,\dots,K\}^N} \sum_{i=1}^{N} F_i(W_{a_i}), \tag{3.2}$$

where W_k denotes the model of cluster k and a_i is the corresponding cluster index of client i.

Different from this paradigm, our method addresses non-IID heterogeneity via layer-wise fuzzy clustering. Instead of forcing the entire model to be assigned to a single cluster, it computes block-wise memberships and performs membership-weighted aggregation for each block. The overview of the proposed layer-wise FCFL framework is shown in Figure 1.

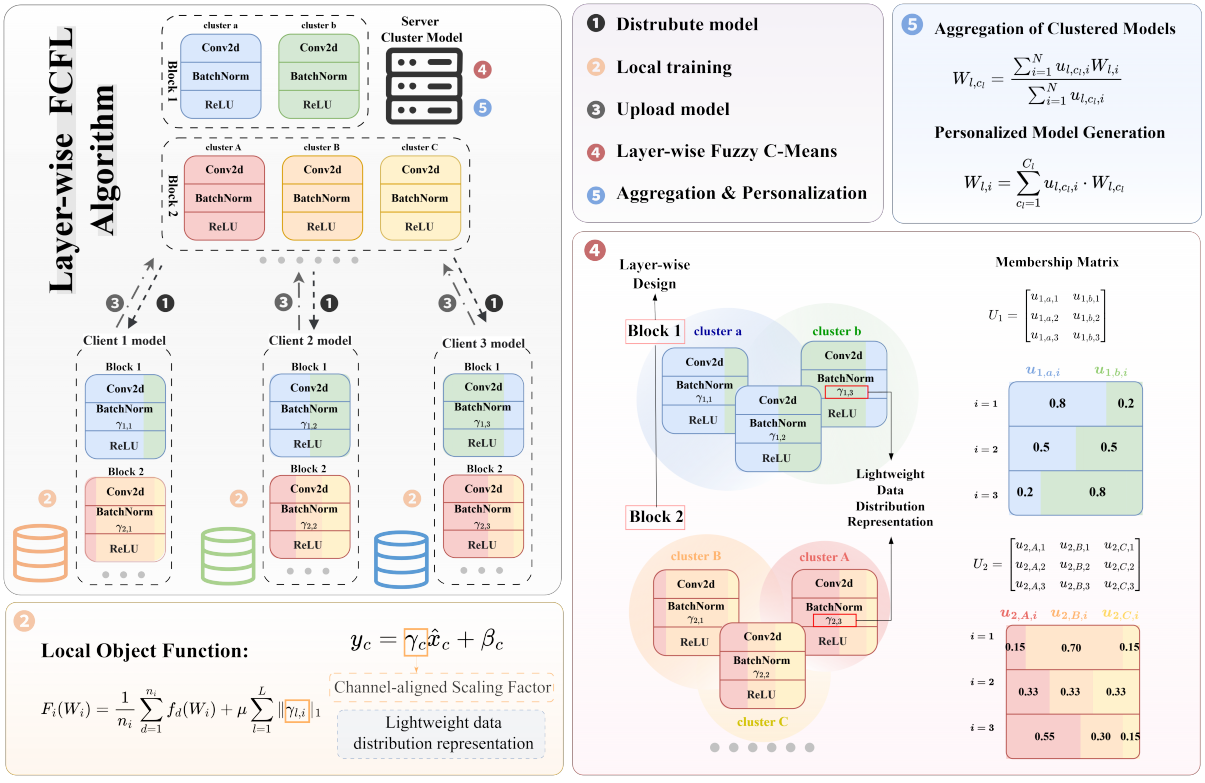


Figure 1. Overview of the proposed layer-wise FCFL framework.

This problem can be decomposed into the following key tasks. These three tasks correspond directly to the local-update, layer-wise fuzzy clustering, aggregation, and personalization steps summarized in Algorithms 1 and 2.

1) Local model training: During local training, each client optimizes its model on $\mathcal{D}_i$ with an additional ℓ_1 penalty on the BN scaling factors $\gamma_{l,i}$, which encourages sparsity and improves their discriminability in characterizing local data distributions. After training, the server extracts $\gamma_{l,i}$ from each client as a lightweight feature representation for layer-wise clustering.
2) Layer-wise FCM: The server performs layer-wise FCM in each communication round to obtain the membership matrix U_l that groups clients with similar data distributions.
3) Membership-aware aggregation and personalization: For each block l and cluster c_l, the server maintains a cluster model W_{l,c_l} and updates it by aggregating client models $W_{l,i}$ with the corresponding fuzzy memberships $u_{l,c_l,i}$. Subsequently, the server fuses these cluster models according to U_l to construct and dispatch personalized models $\{W_{l,i}\}$ for all clients.

4. Layer-wise FCFL algorithm

In this section, we present the overall workflow of the proposed method and summarize the key steps executed on the server and clients. Specifically, we introduce how clients update local models with the sparsity-regularized objective, how the server performs layer-wise fuzzy clustering using BN scaling factors as lightweight representations, and how cluster-aware aggregation and model dispatch are conducted across communication rounds. We also clarify the main design and implementation details of the proposed framework.

4.1. Lightweight data-distribution representation based on scaling factors

To effectively represent client-side data distributions under non-IID settings, our method introduces BN-layer scaling factors as a lightweight proxy representation of client data distributions, enabling precise characterization of local distributional features with extremely low parameter overhead. The scaling factors constitute a set of learnable scalar parameters that are in one-to-one correspondence with specific network units, such as convolutional channels or fully connected neurons.

In our method, these scaling factors are essentially the trainable affine transformation parameters γ embedded in batch normalization layers of deep neural networks. They form a scalar parameter set that is strictly aligned, one-to-one, with network feature-extraction units. Specifically, given the BN layer input x, the layer first normalizes it using mini-batch statistics μ_B and σ_B^2:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}. \tag{4.1}$$

Subsequently, the output is reconstructed via the affine transformation:

$$y = \gamma\hat{x} + \beta, \tag{4.2}$$

where β denotes the shift parameter and ϵ is a constant for numerical stability.

In convolutional neural networks (CNNs), BN layers are typically deployed adjacent to convolutional or fully connected layers. Their core role is to normalize and linearly reconstruct the output features of preceding layers. For convolutional layers, each BN scaling factor γ_c is uniquely associated with one output channel of the preceding convolutional layer, with exactly matched cardinality. This channel-wise relationship is formulated as:

$$y_c = \gamma_c\hat{x}_c + \beta_c. \tag{4.3}$$

The magnitude of γ_c quantifies both the activation strength of that channel and its task-relevant importance under the client's local data distribution, a property that is also leveraged in network slimming [19]. For fully connected layers, each BN scaling factor γ corresponds one-to-one to a neuron in the preceding fully connected layer, representing the contribution of that neuron's output feature to fitting local data. Using scaling factors as the data-distribution representation has several advantages.

1) Implicit semantic correlation: Each scaling factor is directly associated with a specific feature channel/neuron. Its magnitude implicitly reflects the importance or activation strength of that feature on the client's local data, thereby capturing semantic characteristics of the data distribution.
2) High discriminability: By imposing an ℓ_1 regularization term on scaling factors, the training process naturally drives them toward sparsity and polarization (some approaching 0 while others remain significantly nonzero), which yields a more separable feature space.
3) Training stability and compatibility: Scaling factors are jointly optimized with network weights, updated stably, and evolve adaptively during training.
4) Negligible overhead: As a low-dimensional scalar representation, scaling factors incur negligible computation and communication overhead and can be generally applied to various network architectures. Specifically, a BatchNorm layer contains only N_{out} trainable parameters (scaling factors), whereas the corresponding convolutional layer contains $N_{\text{out}} \times N_{\text{in}} \times K \times K$ parameters, reducing the parameter size by several orders of magnitude. Compared with directly using high-dimensional, redundant, and noise-sensitive raw model parameter gradients or weights for clustering, scaling factors provide a compact, semantic, and stable client representation.

Specifically, during local training, client i adds an ℓ_1-norm regularization on the selected BN scaling factors to highlight the differences among clients:

$$F_i(W_i) = \frac{1}{n_i} \sum_{d=1}^{n_i} f_d(W_i) + \mu \sum_{l=1}^{L} \|\gamma_{l,i}\|_1. \tag{4.4}$$

4.2. Layer-wise clustering and aggregation

We combine clustering with the layer-wise structure of deep neural networks and perform clustering and aggregation independently for each network block. This design is motivated by the fact that CNN layers learn features at different abstraction levels: shallow layers capture generic low-level patterns (edges, textures, and colors), while deeper layers become increasingly semantic and class-specific, as shown in [20, 21].

Taking VGG16 as an example, we partition the CNN into L network blocks (VGG16 has $L = 5$ convolutional blocks). For the l-th block ($l = 1, \ldots, L$), we cluster N clients using the scaling factors from the last BN layer of that block. For hard clustering, let $c'_{l,i} \in \{1, \ldots, C_l\}$ denote the assigned cluster index of client i at block l. The block-wise cluster aggregation is defined as

$$W_{l,c_l} = \frac{\sum_{i=1}^{N} \mathbb{1}(c'_{l,i} = c_l) \cdot W_{l,i}}{\sum_{i=1}^{N} \mathbb{1}(c'_{l,i} = c_l)}. \tag{4.5}$$

Equation (4.5) averages the client models assigned to cluster c_l at block l, yielding the cluster-specific parameters W_{l,c_l}. Each client then receives a personalized block model by copying the parameters of its

assigned cluster, i.e.,

$$W_{l,i} = W_{l,c'_{l,i}}. \tag{4.6}$$

The above two equations formulate the rules for the hard-clustering block aggregation and personalization. We then generalize them to soft memberships so that each client can contribute to multiple clusters with different weights.

4.3. Fuzzy clustering

Conventional clustered FL usually adopts hard clustering, where each client is assumed to belong to a unique cluster. As a result, clients near cluster boundaries cannot be fully utilized, and small clusters may suffer from under-training. To address this issue, we adopt FCM, where each client is associated with every cluster through a membership $u_{l,c_l,i} \in [0,1]$ satisfying $\sum_{c_l=1}^{C_l} u_{l,c_l,i} = 1$. Weighted by these memberships, the server aggregates local client models. Personalized models are generated by softly combining the resulting cluster models. In each communication round, the server runs layer-wise FCM on the latest BN scaling factors to update $U_l = [u_{l,c_l,i}]_{i=1,\dots,N,\, c_l=1,\dots,C_l}$ so that cluster assignments can adapt to evolving client representations during training.

With the ℓ_1-norm regularization described in Section 4.1, each client's scaling factors form a high-dimensional sparse vector. To obtain clusters, instead of the Euclidean norms, here the cosine distance $d(x,v) = 1 - \langle x, v\rangle$ is used to measure the difference of the resulting scaling factors in FCM. Since cosine similarity is sensitive to directions compared to norms, it is more suitable here and has also been widely used for clustering large sparse data [22].

As the existing FCM algorithm is derived using Euclidean norms [23], a modified version using cosine distance is provided below.

Let $\{\gamma_{l,i}\}_{i=1}^N$ be the feature vectors of block l from N clients, C_l be the number of clusters at block l, and $m > 1$ be the fuzziness factor. For each client i, the memberships satisfy

$$u_{l,c_l,i} \in [0,1], \qquad \sum_{c_l=1}^{C_l} u_{l,c_l,i} = 1. \tag{4.7}$$

For simplicity, each feature vector $\gamma_{l,i}$ is first normalized such that $\|\gamma_{l,i}\| = 1$. Cosine distance between feature vectors and cluster centers can be expressed as

$$d_{l,c_l,i} = d(\gamma_{l,i}, v_{l,c_l}) = 1 - \frac{\gamma_{l,i}^\top v_{l,c_l}}{\|\gamma_{l,i}\|\,\|v_{l,c_l}\|}.$$

The membership matrix $U_l = [u_{l,c_l,i}]$ and center set $V_l = \{v_{l,c_l}\}_{c_l=1}^{C_l}$ are obtained to solve the following minimization problem at block l:

$$\min_{U_l,V_l} J_l(U_l,V_l) = \sum_{i=1}^N \sum_{c_l=1}^{C_l} (u_{l,c_l,i})^m d_{l,c_l,i} = \sum_{i=1}^N \sum_{c_l=1}^{C_l} (u_{l,c_l,i})^m \left(1 - \frac{\gamma_{l,i}^T v_{l,c_l}}{\|\gamma_{l,i}\|\,\|v_{l,c_l}\|}\right) \quad \text{s.t.} \quad (4.7). \tag{4.8}$$

Following [23], the above optimization problem is solved by alternately solving for minimizer U_l (with V_l fixed) and V_l (with U_l fixed), referred to as the Picard iteration in [23].

Algorithm 1 Layer-wise FCM using cosine distance

Input: unit-normalized data vectors $\{x_i\}_{i=1}^N$ (i.e., $\|x_i\| = 1$), number of clusters C, fuzzifier $m \geq 1$, tolerance ε, maximum iterations $K_{\max}$.
Initialize membership matrix $U^{(0)} = [u_{c_l,i}^{(0)}]$
Randomly initialize cluster centers $V^{(0)} = \{v_1^{(0)}, \ldots, v_C^{(0)}\}$ as unit vectors.
Set iteration counter $k = 0$.
repeat
    $k \leftarrow k + 1$
    for each client i and each cluster c_l **do**
        Compute cosine distance $d(x_i, v_{c_l}^{(k-1)}) = 1 - \langle x_i, v_{c_l}^{(k-1)} \rangle$.
    end for
    if $m = 1$ **then**
        For each client i, define nearest-center index $c_i^\star = \arg\min_{j \in \{1,\ldots,C\}} d(x_i, v_j^{(k-1)})$.
        Set hard memberships:

$$u_{c_l,i}^{(k)} = \begin{cases} 1, & c_l = c_i^\star, \\ 0, & c_l \neq c_i^\star. \end{cases}$$

    else
        Update memberships:

$$u_{c_l,i}^{(k)} = \frac{d(x_i, v_{c_l}^{(k-1)})^{-\frac{1}{m-1}}}{\sum_{c_l'=1}^{C} d(x_i, v_{c_l'}^{(k-1)})^{-\frac{1}{m-1}}}$$

    end if
    Update cluster centers $V^{(k)}$:

$$v_{c_l}^{(k)} = \frac{\sum_{i=1}^N (u_{c_l,i}^{(k)})^m \cdot x_i}{\left\| \sum_{i=1}^N (u_{c_l,i}^{(k)})^m \cdot x_i \right\|}$$

until $\|U^{(k)} - U^{(k-1)}\| < \varepsilon$ **or** $k \geq K_{\max}$
return $U^{(k)}$ (and $V^{(k)}$).

Update of the membership matrix U_l (with V_l fixed) To get the minimizer U_l of (4.8) with fixed V_l, we construct the Lagrangian for each $i = 1, \ldots, N, l = 1, \ldots, L$ as follows:

$$\mathcal{L}_{l,i} = \sum_{c_l=1}^{C_l} (u_{l,c_l,i})^m d_{l,c_l,i} + \lambda_{l,i} \left(\sum_{c_l=1}^{C_l} u_{l,c_l,i} - 1 \right).$$

By setting its derivative equal to zero, we obtain the first-order necessary condition of the optimal U_l:

$$\frac{\partial \mathcal{L}_{l,i}}{\partial u_{l,c_l,i}} = m(u_{l,c_l,i})^{m-1} d_{l,c_l,i} + \lambda_{l,i} = 0,$$

which yields $u_{l,c_l,i} = \left(-\frac{\lambda_{l,i}}{m}\right)^{\frac{1}{m-1}} d_{l,c_l,i}^{-\frac{1}{m-1}}$. Since $\sum_{c_l=1}^{C_l} u_{l,c_l,i} = 1$, we can get $\left(-\frac{\lambda_{l,i}}{m}\right)^{\frac{1}{m-1}} = \frac{1}{\sum_{j=1}^{C_l} d_{l,j,i}^{-\frac{1}{m-1}}}$, and then

$$u_{l,c_l,i} = \frac{d_{l,c_l,i}^{-\frac{1}{m-1}}}{\sum_{j=1}^{C_l} d_{l,j,i}^{-\frac{1}{m-1}}} = \frac{\left(1 - \frac{\gamma_{l,i}^\top v_{l,c_l}}{\|\gamma_{l,i}\| \, \|v_{l,c_l}\|}\right)^{-\frac{1}{m-1}}}{\sum_{j=1}^{C_l} \left(1 - \frac{\gamma_{l,i}^\top v_{l,j}}{\|\gamma_{l,i}\| \, \|v_{l,j}\|}\right)^{-\frac{1}{m-1}}},$$

which is indeed a valid solution satisfying (4.7).

Update of the center set V_l (with U_l fixed) For a fixed cluster c_l, the objective function in (4.8) reduces to

$$J_{l,c_l} = \sum_{i=1}^{N} (u_{l,c_l,i})^m \left(1 - \frac{\gamma_{l,i}^\top v_{l,c_l}}{\|\gamma_{l,i}\| \, \|v_{l,c_l}\|}\right) = \sum_{i=1}^{N} (u_{l,c_l,i})^m - \frac{1}{\|v_{l,c_l}\|} \sum_{i=1}^{N} (u_{l,c_l,i})^m \frac{\gamma_{l,i}^\top v_{l,c_l}}{\|\gamma_{l,i}\|}.$$

Let $b = \sum_{i=1}^{N} (u_{l,c_l,i})^m \frac{\gamma_{l,i}}{\|\gamma_{l,i}\|} = \sum_{i=1}^{N} (u_{l,c_l,i})^m \gamma_{l,i}$ ($\|\gamma_{l,i}\| = 1$), and we can obtain

$$J_{l,c_l} = \text{const} - \frac{b^\top v_{l,c_l}}{\|v_{l,c_l}\|}.$$

Thus minimizing J_{l,c_l} is equivalent to maximizing $\frac{b^\top v_{l,c_l}}{\|v_{l,c_l}\|}$. By the Cauchy–Schwarz inequality, we have $\frac{b^\top v}{\|v\|} \leq \|b\|$ with equality if and only if (iff) v is parallel to b. Therefore, we can obtain

$$v_{l,c_l} = \frac{b}{\|b\|} = \frac{\sum_{i=1}^{N} (u_{l,c_l,i})^m \gamma_{l,i}}{\left\|\sum_{i=1}^{N} (u_{l,c_l,i})^m \gamma_{l,i}\right\|}.$$

Hence, FCM using cosine distance alternates between the above two updates of U_l and V_l until convergence. The FCM procedure is summarized in Algorithm 1.

By implementing the above algorithm, soft membership assignments U_l will be returned. Accordingly, Algorithm 2 uses membership-weighted aggregation. The cluster aggregation and personalized model generation for clients are given as follows, respectively:

$$W_{l,c_l} = \frac{\sum_{i=1}^{N} u_{l,c_l,i} \cdot W_{l,i}}{\sum_{i=1}^{N} u_{l,c_l,i}}; \tag{4.9}$$

$$W_{l,i} = \sum_{c_l=1}^{C_l} u_{l,c_l,i} \cdot W_{l,c_l}. \tag{4.10}$$

Equation (4.9) highlights the role of membership $u_{l,c_l,i}$ as a continuous aggregation weight. Clients with larger memberships contribute more to the corresponding cluster model, enabling more cross-cluster information sharing. Similarly, Equation (4.10) performs membership-weighted fusion across multiple cluster models for personalization. The overall procedure is summarized in Algorithm 2.

Algorithm 2 Personalized FL via layer-wise FCM

Initialization: The cloud server initializes a global model W containing L network blocks and sends W to all clients.
for each training round **do**
  for each client i **do**
    Client i trains on $\mathcal{D}_i$ by minimizing (4.4) for several local epochs and uploads the updated model to the server.
  end for
  The cloud server collects all client models and extracts $\gamma_{l,i}$ for each network block l.
  for each network block l **do**
    Normalize scaling factors $\gamma_{l,i} \leftarrow \gamma_{l,i}/\|\gamma_{l,i}\|$.
    Construct $\Gamma_l \leftarrow \{\gamma_{l,i}\}_{i=1}^{N}$ and update $U_l \leftarrow \mathrm{FCM}(\Gamma_l, C_l, m)$ ($m = 1$ corresponds to hard clustering).
  end for
  for each network block l **do**
    if $m = 1$ **then**
      For each client i, set the hard cluster index $c'_{l,i} = \arg\max_{c_l} u_{l,c_l,i}$.
      Hard-cluster aggregation and personalization:

$$W_{l,c_l} = \frac{\sum_{i=1}^{N} \mathbb{1}(c'_{l,i} = c_l)\, W_{l,i}}{\sum_{i=1}^{N} \mathbb{1}(c'_{l,i} = c_l)};\ W_{l,i} = W_{l,c'_{l,i}}.$$

    else
      Fuzzy aggregation and personalization:

$$W_{l,c_l} = \frac{\sum_{i=1}^{N} u_{l,c_l,i}\, W_{l,i}}{\sum_{i=1}^{N} u_{l,c_l,i}};\ W_{l,i} = \sum_{c_l=1}^{C_l} u_{l,c_l,i}\, W_{l,c_l}.$$

    end if
  end for
  Combine all L personalized model blocks to obtain the complete model W_i for client i.
  The cloud server sends W_i to the corresponding client.
  if the preset number of rounds or accuracy is reached **then**
    Terminate training
  end if
end for
return W_i.

It is noted that for $m = 1$, the memberships degenerate to hard assignments [23]. The memberships are hard indicators with $u_{l,c_l,i} \in \{0, 1\}$ and $\sum_{c_l=1}^{C_l} u_{l,c_l,i} = 1$. Let $c'_{l,i} = \arg\max_{c_l} u_{l,c_l,i}$ denote the selected cluster of client i at block l. Then the aggregation and personalization reduce to (4.5) and (4.6), respectively.

Compared with Eq (4.9), Eq (4.5) replaces continuous memberships with binary indicators, so each client contributes to exactly one cluster. Likewise, Eq (4.6) selects a single cluster model for personalization. Therefore, layer-wise fuzzy clustering degenerates into layer-wise hard clustering, where each client is assigned to a single cluster at each block.

In each communication round, clients perform local training and upload their updated models. The server extracts the BN scaling factors $\gamma_{l,i}$ for each network block, then runs layer-wise FCM to update U_l, and performs membership-weighted aggregation and model dispatch. We use the ℓ_1-regularized scaling factor as the clustering representation because it captures local data characteristics with low overhead and is more stable than whole-model parameters or gradients. It is noted that when client data distributions are static, U_l tends to stabilize as the training progresses. The server may then stop reclustering and fix U_l for the remaining rounds to reduce overhead without sacrificing aggregation quality. If local data distributions drift over time, periodic reclustering should be retained.

Layer-wise cluster numbers $C_l, l = 1 \ldots, L$, as hyperparameters of the algorithm, play an important role in layer-wise FCFL. For appropriate choices of these parameters, a heuristic guideline, increasing the cluster number as the network layer deepens, can be adopted by considering the feature hierarchy of convolutional neural networks. As has been stated in Section 4.2, shallow blocks usually extract generic low-level patterns that can be widely shared across clients, whereas deeper blocks encode more semantic and category-related information and thus require finer-grained client grouping. Therefore, for the first block, generally a smaller cluster number can be assigned, typically $C_1 = 1$, while for the intermediate and last blocks, the number of clusters assigned gradually increases. Furthermore, one can also resort to elbow and silhouette methods [24, 25] to find the optimal cluster numbers in a certain sense. These two commonly used methods try to determine the optimal cluster number for each block by identifying the elbow point and silhouette coefficient, respectively. The elbow point provides a practical trade-off between clustering quality and model complexity, while the silhouette score helps identify cluster numbers with better inter-cluster separability. The appropriate C_l values can then be selected by jointly considering these two methods and the above heuristic guideline, thereby reducing the risk of inappropriate choices of cluster numbers. It is noted that the proposed FCFL usually exhibits high robustness with respect to different cluster numbers, as shown in Section 5.5.

Using the scaling factors as lightweight representations of client data distributions and layer-wise clustering lead to high efficiency of the proposed layer-wise FCFL. A brief analysis of server-side clustering time complexity introduced by layer-wise FCM is provided here. The per-layer run time of K-means and the original FCM of Bezdek et al. [23] are, respectively, $O\left(T_{\text{kmeans}}\, N\, C_l\, d_l\right)$ [26] and $O\left(T_{\text{fcm}}\, N\, C_l^2\, d_l\right)$ [27], where T_{kmeans} and T_{fcm} are iteration numbers of these two algorithms, respectively, and $d_l = \dim(\gamma_l)$ is the dimension of the BN scaling-factor representation at layer l. Compared with K-means, obviously, the FCM imposes higher per-iteration computation. The extra factor of C_l comes from the soft-membership update. However, by combining two updates of the membership and cluster centers into a single update of cluster centers (see Kolen and Hutcheson [27]), the per-layer time complexity of FCM is reduced to $O\left(T_{\text{fcm}}\, N\, C_l\, d_l\right)$, which has the same leading-order dependence on N, C_l, and d_l as K-means while still providing membership information. Therefore, the per-layer runtime of FCM is comparable to that of K-means. Furthermore, in the layer-wise design, the clustering of different layers is mutually independent with no data dependency, and thus can be executed in parallel. Consequently, layer-wise clustering does not impose an additional serial burden on the server. Instead, the wall-clock time of FCM is further reduced by the parallel implementation. Finally, in conventional clustered FL, full-model parameter clustering with K clusters and parameter dimension P has a time complexity of $O(T_{\text{kmeans}}^{\text{full model}} NKP)$. Since the dimension $d_l, l = 1 \ldots, L$, of each BN scaling factor is far smaller than that of the full model parameters for modern CNNs, i.e., $P \gg d_l$, the runtime of layer-wise FCFL is usually significantly shorter than that of conventional full-model clustered FL algorithms. In

conclusion, the low time complexity of per-layer clustering, parallel implementation mechanism, and lightweight representations of client data distributions render high efficiency of the proposed layer-wise FCFL.

5. Performance evaluation

In this section, extensive experiments are conducted to validate the effectiveness and efficiency of the proposed layer-wise FCFL under non-IID data distributions. We compare it with the existing clustered FL baselines such as CFL, IFCA, and FLUX [28] on four datasets (MNIST, EMNIST, CIFAR-10, and CIFAR-100) to demonstrate its performance advantages. In addition, we empirically evaluate the server-side clustering runtime and design a series of ablation studies to verify the effectiveness of the key components in our algorithm design. Finally, robustness of the algorithm with respect to layer-wise cluster numbers is analyzed by comparing its accuracies under different cluster-number settings.

5.1. Experimental setup

Our experimental setup includes (i) the construction of non-IID client datasets, (ii) model architectures, and (iii) training hyperparameters and baselines for comparison.

5.1.1. Non-IID data partitioning

We simulate data heterogeneity by partitioning the training sets of MNIST, EMNIST, CIFAR-10, and CIFAR-100 across clients using a Dirichlet distribution. Specifically, for each class, we sample client-wise class proportions from a Dirichlet distribution Dir(α) to form a proportion matrix, and then allocate samples to clients accordingly. A smaller α yields a more imbalanced class distribution across clients, corresponding to a higher degree of data heterogeneity.

The Dirichlet parameters and client/data configurations are summarized in Table 2.

Table 2. Dirichlet-based non-IID partition settings.

Dataset	Number of classes	α	Number of clients (N)	Number of training samples
MNIST	10	0.1	30	2000
EMNIST	62	0.3	60	4500
CIFAR-10	10	0.1	60	3000
CIFAR-100	100	0.15	270	50,000

5.1.2. Models

We adopt two backbone networks as both local and global models: VGG16 with batch normalization (BN) and a SIMPLE-CNN with two convolutional layers. SIMPLE-CNN is used for MNIST and EMNIST, and VGG16 for CIFAR-10 and CIFAR-100. For VGG16, we perform a layer-wise partition of the feature extractor into five convolutional blocks (each block contains multiple convolutional layers and BN layers). When a block contains multiple BN layers, we use the scaling factor of the last BN layer in that block as the data-distribution representation. Therefore, VGG16 corresponds to five feature clustering layers. Following the intuition that shallow layers learn more general features while deeper layers capture more personalized features, we set the per-block cluster numbers $\{C_l\}_{l=1}^{5}$ to be small in

shallow blocks and larger in deeper blocks. Specifically, we set $\{C_l\}_{l=1}^{5} = [1, 3, 5, 5, 8]$ for the five feature blocks. For SIMPLE-CNN, since it only contains two convolutional layers and has a smaller scale, we partition it into two layers for clustering and similarly adopt a "small-to-large" rule, with $\{C_l\}_{l=1}^{2} = [1, 8]$.

We do not define a separate clustering scheme for the classifier layer. Instead, its clustering and aggregation follow the last feature layer (i.e., it uses the scaling factor of the last BN layer as the data-distribution representation, and the classifier is aggregated according to the cluster assignment of that layer).

5.1.3. Training setting and baselines

For all methods, the corresponding models are trained for 120 global communication rounds with 5 local epochs per round under the same non-IID partitions as in Table 2. Local updates use AdamW with a learning rate of 10^{-3} and a weight decay of 10^{-4}, and the ℓ_1 regularization coefficient on BN scaling factors is set to $\mu = 10^{-5}$. For efficiency, after sufficient local training in the first communication round, fuzzy memberships are estimated once and kept fixed thereafter, since client data partitions remain static during training. For performance comparison, we evaluate layer-wise FCFL with FedAvg, single-client local training, CFL, IFCA, and FLUX under the same experimental setting. For ablation, we compare four configurations combining layer-wise/whole-model clustering with hard/fuzzy assignments (Settings A–D in Section 5.4).

5.2. Effectiveness of clustering representations and efficiency of layer-wise fuzzy clustering

Effectiveness of client representations for clustering. We further investigate, under a highly non-IID setting, the effectiveness of two different client representations as the basis for clustering. The experiment uses 30 clients on CIFAR-10 with a one-class-per-client partition: every 3 clients share data from the same class, covering all 10 classes and forming 10 client groups. After 8 rounds of local training, we extract two types of representations: (1) BN scaling factors and (2) whole model parameters. We perform k-means clustering with $k = 10$ using cosine distance for (1) and Euclidean distance (ℓ_2) for (2), and evaluate clustering agreement against the ground-truth class-based grouping using adjusted Rand index (ARI) and normalized mutual information (NMI). We also present the representation dimensionality and runtime. The results are summarized in Table 3: BN scaling factor-based clustering not only achieves higher clustering quality than full-model-parameter-based clustering but also provides an approximately 1000× improvement in computational efficiency, demonstrating superior performance in both clustering accuracy and efficiency.

Table 3. Clustering quality and efficiency of different client representations on CIFAR-10.

Method	ARI	NMI	Dim.	Runtime (s)
Scaling factor	1.000	1.000	512	0.028
Model parameters	0.812	0.935	14,724,042	76.669

Empirical runtime of layer-wise fuzzy clustering. To verify the efficiency of the proposed layer-wise FCFL, a comparison of server-side clustering runtimes of different clustering strategies is made on CIFAR-10 under the same settings as in Table 2. As shown in Table 4, average runtimes (over 10 clustering runs) of both layer-wise hard clustering and layer-wise soft clustering are significantly

shorter than that of full-model-parameter hard clustering, as the dimension $d_l, l = 1 \ldots, L$, of each BN scaling factor is far smaller than that of the full model parameters. Using the full-model-parameter hard clustering scheme as the baseline, the proposed BN-based layer-wise hard clustering takes only 8.7 ms, achieving a 284.1× speedup over full-model K-means. Our BN-based layer-wise FCM has a runtime of 63.7 ms, higher than that of layer-wise hard clustering due to more iterations. However, just as discussed in Section 4.3, the per-iteration runtime of 0.123 ms (taking Feature.41, for example) of the former is only slightly longer than that of 0.113 ms of the latter, and even the improved implementation of FCM [27] has not been utilized. It is noted that compared with layer-wise FCM, layer-wise hard clustering discards soft membership information and yields lower accuracy of the corresponding algorithm in our ablation study as shown below. These results verify that layer-wise fuzzy clustering achieves high efficiency while retaining fine-grained soft assignments.

Table 4. Server-side clustering runtime comparison (CIFAR-10, 60 clients, $\alpha = 0.1$).

Clustering scheme	Runtime (ms)	Feature.41 iter. (ms)	Speedup
Full model parameters + hard clustering (K-means)	2,471.3 ± 2.3	–	Baseline
BN scaling factors + layer-wise hard clustering (K-means)	8.7 ± 0.9	0.113 ± 0.007	284.1×
BN scaling factors + layer-wise soft clustering (FCM)	63.7 ± 0.9	0.123 ± 0.059	38.8×

5.3. Accuracy comparison

To evaluate the effectiveness of the proposed method, we conduct comparative experiments on four widely used federated learning datasets, i.e., MNIST, EMNIST, CIFAR-10, and CIFAR-100, against FedAvg, the single-client local training, CFL, IFCA, and FLUX baselines. In the experiments, all methods are evaluated under identical communication rounds, local epochs, and hyperparameter settings; test accuracy serves as the primary metric. Table 5 summarizes the average test accuracies over the last 10 communication rounds, and Figure 2 visualizes the full accuracy trajectories across the four datasets. Overall, the results show that our method consistently outperforms or is comparable to existing approaches under all four non-IID settings.

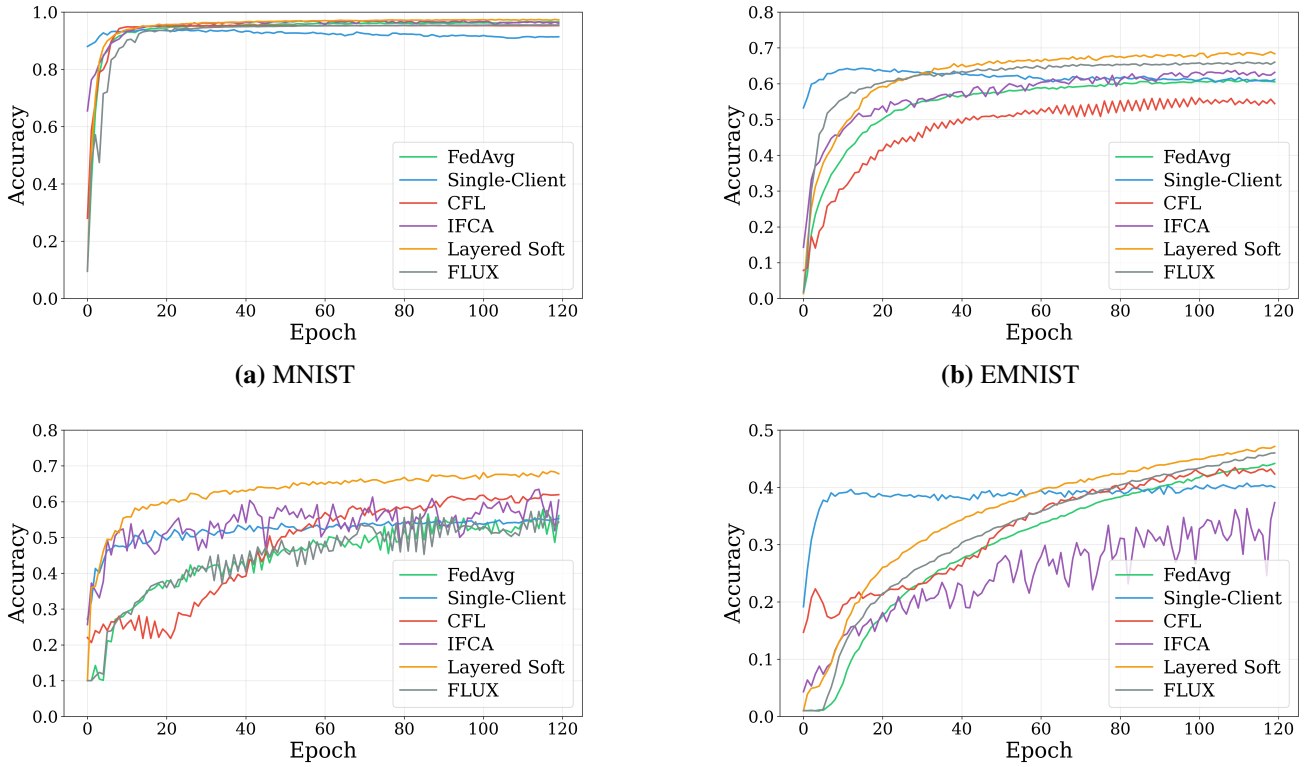
Table 5. Average test accuracy (%) over the last 10 rounds on MNIST, EMNIST, CIFAR-10, and CIFAR-100.

Method	MNIST	EMNIST	CIFAR-10	CIFAR-100
FedAvg	96.57	60.75	54.12	43.58
CFL	95.27	54.86	61.43	42.80
IFCA	96.38	62.71	58.95	32.31
Single-Client	91.36	61.01	54.93	40.27
FLUX	95.49	65.78	54.51	44.06
Layer-wise FCFL	**97.56**	**68.42**	**68.53**	**46.67**

As summarized in Table 5, layer-wise FCFL achieves the highest accuracy on all datasets. Compared with the strongest baseline on each dataset, it improves accuracy by 0.99% (MNIST), 2.64% (EMNIST), 7.10% (CIFAR-10), and 2.61% (CIFAR-100), respectively.

Figure 2 shows that layer-wise FCFL reaches high accuracy earlier and maintains more stable performance in later communication rounds. On MNIST, where accuracies are already close to saturation

(above 95% for most methods), the average gain over the strongest baseline is modest (+0.99%), as reported in Table 5. In contrast, the improvements on EMNIST, CIFAR-10, and CIFAR-100 are substantially larger (+2.64%, +7.10%, and +2.61%, respectively). Overall, these results suggest that the proposed method benefits both convergence behavior and final accuracy, especially on more challenging benchmarks.



(a) MNIST **(b)** EMNIST

(c) CIFAR-10 **(d)** CIFAR-100

Figure 2. Accuracy comparison on four datasets.

5.4. Ablation study

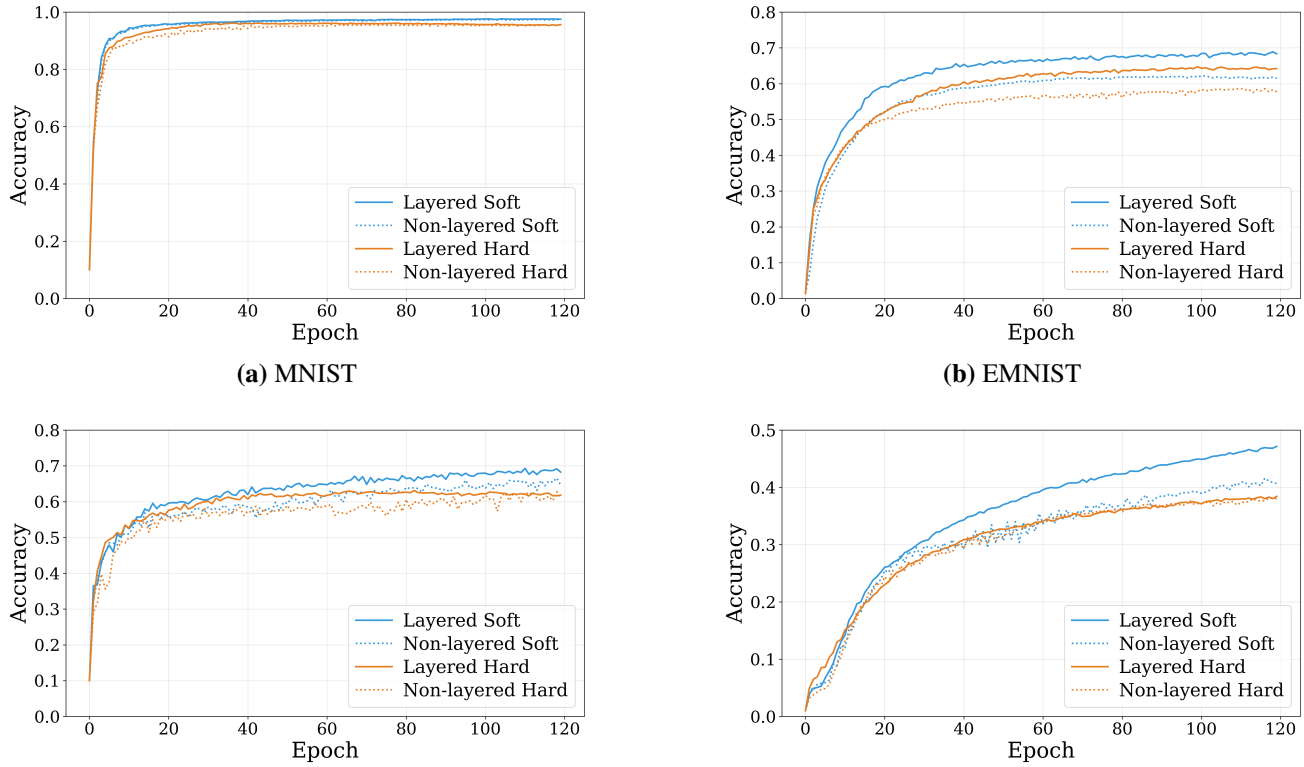
Table 6. Average test accuracy (%) over the last 10 rounds under different ablation settings.

Method	MNIST	EMNIST	CIFAR-10	CIFAR-100
Setting A	95.39	58.14	61.54	37.76
Setting B	97.31	67.94	65.11	41.39
Setting C	95.49	64.27	61.99	38.13
Setting D	**97.56**	**68.42**	**68.53**	**46.67**

To analyze the contributions of layer-wise clustering and fuzzy clustering to the overall performance, we design the following four configurations for ablation experiments:

- **Setting A:** non-layer-wise + hard clustering
- **Setting B:** non-layer-wise + fuzzy clustering
- **Setting C:** layer-wise + hard clustering (the special case of our method ($m = 1$))
- **Setting D:** layer-wise + fuzzy clustering (our method)

We compare all four settings under an identical data partition, the same number of communication rounds, and the same local training configuration. Test accuracy is used as the evaluation metric. Table 6 presents the average test accuracy over the last 10 rounds, while Figure 3 visualizes the performance trajectories across datasets. Overall, either layer-wise clustering or fuzzy clustering improves performance, and combining them yields the best results.



(a) MNIST **(b)** EMNIST

(c) CIFAR-10 **(d)** CIFAR-100

Figure 3. Ablation study results on four datasets.

Quantitatively, fuzzy clustering is beneficial with and without layer-wise partitioning. Without layer-wise partitioning, Setting B improves over Setting A by +1.92%, +9.80%, +3.57%, and +3.63% on MNIST, EMNIST, CIFAR-10, and CIFAR-100, respectively. The effect becomes more pronounced under layer-wise partitioning: switching from hard to fuzzy assignment (Setting D vs. Setting C) yields especially large gains on CIFAR-10 and CIFAR-100 (+6.54% and +8.54%). This synergy suggests that fuzzy memberships complement layer-wise modeling rather than substituting for it. Consistent with this observation, Figure 3 shows that Setting D dominates the other variants throughout the communication rounds, highlighting the importance of combining both designs to realize the full performance benefits.

5.5. Robustness under different cluster-number settings

To verify the robustness of the proposed FCFL under different cluster-number settings, we conduct extensive experiments with various optimized, heuristic, and random configurations on CIFAR-10 using the non-IID partition specified in Table 2. In addition to the heuristic cluster number setting $[1, 3, 5, 5, 8]$ described in Section 4.3, we also evaluate the setting $[6, 9, 10, 9, 9]$ using the elbow method, the setting $[2, 2, 2, 2, 10]$ using the silhouette method, two additional heuristic settings $[1, 5, 8, 8, 15]$,

[1, 7, 15, 15, 30], and a completely random setting [12, 7, 25, 3, 18]. These configurations include not only optimal choices (in some sense), but also heuristic empirical choices and highly irregular choices. The resulting comparison is shown in Table 7 and Figure 4.

Table 7. Accuracy comparison across clustering strategies and layer-wise cluster configurations.

Clustering strategy	Cluster number configuration	Average accuracy of last 10 rounds (%)	Final round accuracy (%)
Soft clustering	[1, 3, 5, 5, 8] (Heuristic)	68.55	68.30
Soft clustering	[6, 9, 10, 9, 9] (Using the elbow method)	67.81	67.80
Soft clustering	[2, 2, 2, 2, 10] (Using the silhouette coefficient)	67.06	67.30
Soft clustering	[1, 5, 8, 8, 15] (Heuristic)	65.69	65.50
Hard clustering	[2, 2, 2, 2, 10] (Using the silhouette coefficient)	65.08	65.10
Soft clustering	[12, 7, 25, 3, 18] (Completely random)	64.78	64.90
Soft clustering	[1, 7, 15, 15, 30] (Heuristic)	63.02	63.40
Hard clustering	[1, 3, 5, 5, 8] (Heuristic)	61.99	61.80
Hard clustering	[6, 9, 10, 9, 9] (Using the elbow method)	60.96	61.20
Hard clustering	[1, 5, 8, 8, 15] (Heuristic)	60.61	60.00
Hard clustering	[12, 7, 25, 3, 18] (Completely random)	59.44	59.50
Hard clustering	[1, 7, 15, 15, 30] (Heuristic)	59.17	59.50
IFCA	Baseline	58.95	60.43

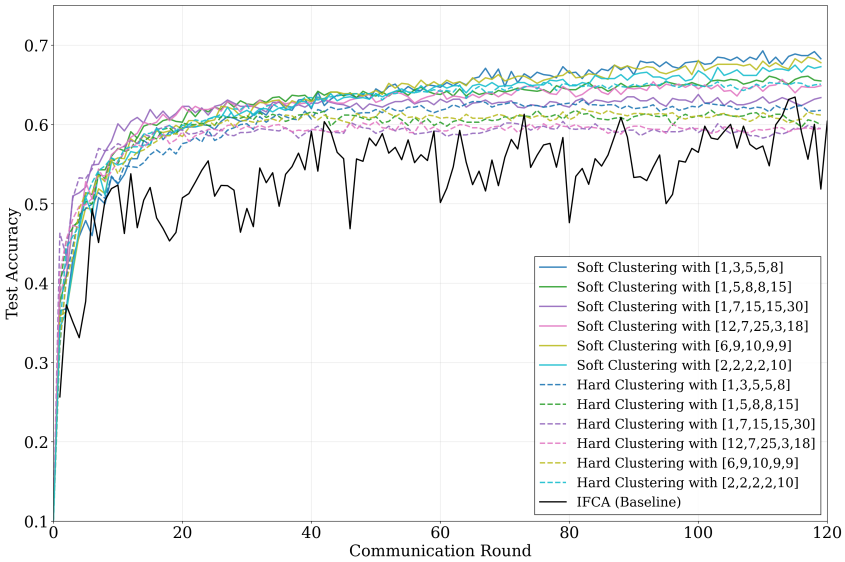


Figure 4. Accuracy comparison under different layer-wise cluster-number settings.

As shown in Table 7 and Figure 4, even when the cluster number of some blocks is heuristically arbitrarily selected or is randomly specified, layer-wise FCFL still maintains stable performance and remains competitive with the baseline methods. More importantly, the layer-wise soft clustering consistently shows stronger robustness than layer-wise hard clustering under different C_l settings. This is because fuzzy memberships allow each client to learn information from multiple cluster models, thereby mitigating the negative effect of occasional cluster-number mismatch in some individual layers.

These observations confirm that the proposed method is not excessively dependent on any particular C_l configuration and has favorable robustness with respect to cluster-number selection.

Finally, although we have evaluated our method only on SIMPLE-CNN and VGG16, the proposed layer-wise FCFL exhibits inherent scalability to deeper and more complex networks. Since BN scaling factor-based representations scale linearly with network depth, our method, when it scales to larger architectures, avoids the dimensionality explosion inherent in full-parameter clustering. Extending the proposed layer-wise FCFL to large-scale modern architectures (e.g., deeper residual networks (ResNets)) to further validate its efficacy in complex tasks remains an important direction for future research.

6. Conclusions

This paper proposes layer-wise FCFL to mitigate the performance degradation of conventional FL under non-IID data distributions. Instead of clustering and aggregating the whole model, we leverage the scaling factors of batch normalization (BN) layers as lightweight client representations, and further enhance their discriminability by adding an ℓ_1 sparsity regularization term in the local objective. Motivated by fuzzy clustering and the layer-wise nature of deep networks, we conduct layer-wise clustering and layer-wise aggregation so that shallow layers can better share general features while deeper layers can preserve more personalized knowledge. Extensive experiments on four datasets (MNIST, EMNIST, CIFAR-10, and CIFAR-100) show that the proposed method achieves remarkable improvements in accuracy and training stability compared with representative clustered FL baselines such as CFL and IFCA. Ablation studies further confirm the effectiveness of the layer-wise design and the fuzzy aggregation strategy.

Future work can be pursued in the following directions:

- **Impact of different partition schemes:** Different model architectures (or different ways of partitioning the same model) may imply different levels of abstraction and parameter-sharing needs. It is worth investigating how alternative partition strategies (e.g., by convolutional blocks, residual groups, or functional modules such as attention/multilayer perceptron(MLP)) affect clustering quality and final accuracy, and designing more general partition guidelines accordingly.
- **Robustness under stringent federated conditions:** Future work can extend the layer-wise fuzzy clustering framework to diverse tasks and rigorously evaluating its robustness under partial client participation, severe communication constraints, and noisy or adversarial updates.

Use of AI tools declaration

The authors declare that no generative-AI tools were used in the writing or research of this manuscript.

Acknowledgments

This work is supported in part by the Shandong Provincial Natural Science Foundation (Grant Nos. ZR2025MS1105, ZR2026LZJ005), Shandong Provincial Postdoctoral Innovation Project (Grant No. 202003005), Shandong Provincial Key Research and Development Program (Grant No. 2024CXPT052), Qingdao Municipal Bureau of Science and Technology (Grant No. 26-1-5-cspz-25-nsh), and Fundamental Research Funds for the Central Universities (Grant No. 14380002).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. Intersoft Consulting, *General Data Protection Regulation (GDPR)*, 2018. Available from: https://gdpr-info.eu/.
2. N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, et al., The future of digital health with federated learning, *npj Digital Med.*, **3** (2020), 119. https://doi.org/10.1038/s41746-020-00323-1
3. P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, et al., Advances and open problems in federated learning, *Found. Trends Mach. Learn.*, **14** (2021), 1–210. https://doi.org/10.1561/2200000083
4. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in *Artificial Intelligence and Statistics*, (2017), 1273–1282.
5. A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, et al., Federated learning for mobile keyboard prediction, preprint, arXiv:1811.03604.
6. Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, preprint, arXiv:1806.00582.
7. F. Sattler, K. Müller, W. Samek, Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints, *IEEE Trans. Neural Networks Learn. Syst.*, **32** (2020), 3710–3722. https://doi.org/10.1109/TNNLS.2020.3015958
8. A. Ghosh, J. Chung, D. Yin, K. Ramchandran, An efficient framework for clustered federated learning, in *Advances in Neural Information Processing Systems*, **33** (2020), 19586–19597.
9. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, in *Proceedings of Machine Learning and Systems*, **2** (2020), 429–450.
10. M. G. Arivazhagan, V. Aggarwal, A. K. Singh, S. Choudhary, Federated learning with personalization layers, preprint, arXiv:1912.00818.
11. T. Li, S. Hu, A. Beirami, V. Smith, Ditto: Fair and robust federated learning through personalization, in *International Conference on Machine Learning*, (2021), 6357–6368.
12. Z. Zhu, J. Hong, J. Zhou, Data-free knowledge distillation for heterogeneous federated learning, in *International Conference on Machine Learning*, (2021), 12878–12889.
13. Y. Jiang, S. Wang, V. Valls, B. J. Ko, W. Lee, K. K. Leung, et al., Pfa: Privacy-preserving federated adaptation for effective model personalization, in *Proceedings of the 2021 ACM SIGCOMM Workshop on Network Meets AI & ML (NetAI)*, (2021), 61–67.
14. Y. Kim, E. Al Hakim, J. Haraldson, H. Eriksson, J. M. B. da Silva, C. Fischione, Dynamic clustering in federated learning, in *ICC 2021-IEEE International Conference on Communications*, IEEE, (2021), 1–6. https://doi.org/10.1109/ICC42927.2021.9500877

15. B. Liu, Y. Cai, Z. Zhang, Y. Li, L. Wang, D. Li, et al., Distfl: Distribution-aware federated learning for mobile scenarios, *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, **5** (2021), 1–26. https://doi.org/10.1145/3494966

16. E. Yoo, H. Ko, S. Pack, Fuzzy clustered federated learning algorithm for solar power generation forecasting, *IEEE Trans. Emerging Top. Comput.*, **10** (2022), 2092–2098. https://doi.org/10.1109/TETC.2022.3142886

17. C. Li, G. Li, P. K. Varshney, Federated learning with soft clustering, *IEEE Internet Things J.*, **9** (2022), 7773–7782. https://doi.org/10.1109/JIOT.2021.3113927

18. Y. Deng, A. Wang, L. Zhang, Y. Lei, B. Li, Y. Li, FedRFC: Federated learning with recursive fuzzy clustering for improved non-iid data training, *Future Gener. Comput. Syst.*, **160** (2024), 835–843. https://doi.org/10.1016/j.future.2024.06.049

19. Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in *2017 IEEE International Conference on Computer Vision (ICCV)*, (2017), 2755–2763.

20. D. Bau, B. Zhou, A. Khosla, A. Oliva, A. Torralba, Network dissection: Quantifying interpretability of deep visual representations, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2017), 6541–6549.

21. M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in *European Conference on Computer Vision*, Springer International Publishing, Cham, (2014), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

22. I. S. Dhillon, D. S. Modha, Concept decompositions for large sparse text data using clustering, *Mach. Learn.*, **42** (2001), 143–175. https://doi.org/10.1023/A:1007612920971

23. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Springer Science & Business Media, 2013.

24. R. L. Thorndike, Who belongs in the family, *Psychometrika*, **18** (1953), 267–276.

25. P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *J. Comput. Appl. Math.*, **20** (1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

26. J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, *J. R. Stat. Soc. C*, **28** (1979), 100–108. https://doi.org/10.2307/2346830

27. J. F. Kolen, T. Hutcheson, Reducing the time complexity of the fuzzy c-means algorithm, *IEEE Trans. Fuzzy Syst.*, **10** (2002), 263–267. https://doi.org/10.1109/91.995126

28. D. Fenoglio, M. Li, P. Barbiero, N. Lane, M. Langheinrich, M. Gjoreski, Flux: Efficient descriptor-driven clustered federated learning under arbitrary distribution shifts, in *Advances in Neural Information Processing Systems*, **38** (2026), 71229–71292.

AIMS Press