SCALABLE AND DECENTRALIZED TRAINING OF MIXTURE OF EXPERTS WITH COLLABORATIVE GATING

Anonymous authors
Paper under double-blind review

ABSTRACT

Decentralized federated learning with Mixture-of-Experts (MoE) allows clients to directly share lightweight experts and combine them locally, avoiding the server dependency and knowledge degradation of centralized aggregation. However, existing decentralized FL-MoE methods assume fully-connected communication among all clients, which incurs quadratic per-round cost and does not scale. In addition, each client's gate is trained only on local data, which is insufficient to learn reliable routing under non-IID distributions. We present DCGMoE (Decentralized Collaborative Gating MoE), a decentralized FL-MoE framework that addresses both issues. First, DCGMoE introduces a *data-aware grouping* strategy that partitions federations into groups of complementary clients using local statistics such as label distribution. Each client then communicates only with members of its own group, which reduces the per-round communication cost from quadratic to sub-quadratic growth as the federation size increases. Second, DCGMoE proposes a *collaborative gating* mechanism that jointly trains the routing module across clients so that each gate learns to exploit the expert pool within its group. The proposed DCGMoE achieves higher global accuracy than both centralized and decentralized FL baselines on Fashion-MNIST, CIFAR-10, and CIFAR-100 under non-IID partitioning with α=0.3. At a federation of 100 clients, DCGMoE further achieves substantially lower per-round communication cost and shorter per-round training time than baselines.

1 INTRODUCTION

Mixture-of-Experts (MoE) (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Du et al., 2022; Jiang et al., 2024) routes each input to a small set of specialized experts through a learned gate, allowing the model to draw on different parameters for different inputs. A recent line of FL work adopts a typical architecture, where each client's local model (or part of it) is treated as an expert, and a gate is introduced to combine experts across clients (Listo Zec et al., 2020; Guo et al., 2021; Dun et al., 2023; Yi et al., 2026; Mei et al., 2024; Jiang et al., 2025; Liu et al., 2025; Zhang et al., 2026a; Luo et al., 2025; Wang et al., 2025; Zhan et al., 2024). While these methods exploit MoE for expert specialization under non-IID data, they all rely on a central server to aggregate the gate, the experts, or both. The aggregation can damage each client's specialization through parameter averaging, and the server reintroduces a single point of failure and a communication bottleneck.

Decentralized federated learning removes the central coordinator and lets clients communicate directly with one another. In particular, dFLMoE (Xie et al., 2025) extends this idea to MoE, where each client maintains a model architecture mainly composed of a feature extractor (i.e., body), a feature-space transform module (i.e., FST), an expert (i.e., head), and a classifier. Each client transmits its lightweight head to other clients. The client's own head together with the received heads form an expert pool. All experts are then combined locally through a per-client gate. Because experts are exchanged rather than aggregated, no client's knowledge is ever merged into a shared global representation, addressing both the server dependence and the knowledge degradation of FL-MoE in a single design. However, two challenges remain. The framework assumes fully-connected communication among all N clients, which incurs $\mathcal{O}(N^2)$ per-round cost and does not scale. The gate of each client is also trained only on local data, so it cannot learn which peer experts suit inputs beyond its narrow class subset, especially under non-IID distributions.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: **Comparison of DCGMoE with representative federated mixture-of-experts methods.** *Decentralized*: no central coordinator. *Lightweight*: only lightweight modules (e.g., heads, prompts) exchanged per round rather than full models. *Gate training*: *Local-only* = each client trains its gate independently; *Collaborative* = gate parameters are jointly trained across clients (e.g., by aggregation); *N/A* = no routing gate. *Expert sharing*: *Fully shared* = all clients share the full expert pool with synchronized parameters; *Partially shared* = experts are exchanged or only a subset is shared while the rest are private; *Local-only* = each client keeps a private expert. *Communication*: *Central* = clients communicate only through a central server; *All-to-All* = all clients communicate pairwise; *Grouped* = clients are partitioned into groups with intra-group communication only.

Type	Method	Decentralized	Lightweight	Gate training	Expert sharing	Communication
FL-MoE	FedJETs (Dun et al., 2023)	✗	✗	Collaborative	Fully shared	Central
	pFedMoE (Yi et al., 2026)	✗	✓	Local-only	Partially shared	Central
	FedMoE (Mei et al., 2024)	✗	✗	Collaborative	Partially shared	Central
	Fed-MoE (Jiang et al., 2025)	✗	✓	Collaborative	Partially shared	Central
	FLEx (Liu et al., 2025)	✗	✓	N/A	Partially shared	Central
	FLEX-MoE (Zhang et al., 2026a)	✗	✗	Local-only	Fully shared	Central
DFL-MoE	dFLMoE (Xie et al., 2025)	✓	✓	Local-only	Partially shared	All-to-All
	DCGMoE (Ours)	✓	✓	**Collaborative**	Partially shared	**Grouped**

We address both challenges with DCGMoE (Decentralized Collaborative Gating MoE). First, to make decentralized FL-MoE practical at scale, we introduce a *data-aware grouping* strategy that partitions N clients into K groups of complementary clients using their local data statistics such as label distributions. After initialization, all communication and synchronization take place within each group, reducing per-round cost from $\mathcal{O}(N^2)$ to approximately $\mathcal{O}(N^2/K)$ under balanced clustering. In particular, by setting $K = \sqrt{N}$, the per-round cost becomes $\mathcal{O}(N^{1.5})$, which is sub-quadratic in the federation size. Second, to enable cross-client routing knowledge without introducing a server, we propose *collaborative gating*, which synchronizes each group's gate so that it is jointly trained on data from all group members rather than in isolation. As Table 1 summarizes, DCGMoE is the only method that is simultaneously decentralized, lightweight, trains the gate collaboratively, partially shares experts, and scales beyond a full connection through group communication. Our main contributions are as follows:

- We propose DCGMoE, the first decentralized FL-MoE framework that simultaneously scales to large client populations and trains the gate collaboratively across clients.
- We design a *data-aware grouping* mechanism that reduces per-round communication from $\mathcal{O}(N^2)$ to $\mathcal{O}(N^{1.5})$ with $K \approx \sqrt{N}$ groups, enabling decentralized FL-MoE to scale to larger systems, e.g., with N=100 clients or more.
- We propose *collaborative gating* with two complementary synchronization strategies, Avg-Sync and Roll-Sync, that allow each client's gate to leverage knowledge across non-IID peers without a central server.
- Extensive experiments on Fashion-MNIST, CIFAR-10, and CIFAR-100 under non-IID partitioning with α=0.3 show that DCGMoE achieves higher global accuracy than both centralized and decentralized FL baselines, with larger margins on the more difficult datasets.

2 Related Work

Decentralized federated learning. Decentralized FL, also known as peer-to-peer FL (Roy et al., 2019), removes the dependency on a central server by letting clients communicate directly with one another, eliminating the central server as a single point of failure and trust. Most existing work in this space focuses on the communication infrastructure. For instance, integrating FL with blockchain to strengthen trust guarantees (Liu et al., 2020; Agarwal et al., 2024; Qin et al., 2024; Zhao et al., 2021) or improving robustness to system and statistical heterogeneity through client selection and clustering (Li et al., 2024; Chen et al., 2022; Wang et al., 2021; Salmeron et al., 2023). These approaches inherit the parameter-averaging style of centralized FL and therefore still suffer from knowledge damage on non-IID data.

Federated learning with Mixture-of-Experts. A growing body of work integrates MoE (Shazeer et al., 2017) with FL, treating each client's local model (or a part of it) as an expert and introducing

a gate to combine experts across clients. Early works (Listo Zec et al., 2020; Reisser et al., 2021; Guo et al., 2021; Parsaeefard et al., 2021; Isaksson et al., 2023) established the framework for personalization, robustness, and specialization. More recent efforts pursue different design goals: parameter-efficient federated tuning of pretrained MoE LLMs (Wu et al., 2023; Liu et al., 2025; Qi et al., 2024; Bian et al., 2026; Zhang et al., 2026b; Wang et al., 2025), personalization through model-heterogeneous experts and gates (Yi et al., 2026; Mei et al., 2024; Luo et al., 2025), scaling MoE to large server-coordinated federations (Dun et al., 2023; Jiang et al., 2025; Zhang et al., 2026a), and continual adaptation under distribution shift (Bhope et al., 2025). FedMoE-DA (Zhan et al., 2024) introduces peer-to-peer synchronization between clients to reduce server traffic. Despite differences in scope, these methods rely on a central server to aggregate the gate, the experts, or both, inheriting the communication bottleneck and trust concentration of centralized FL.

Decentralized federated MoE. Combining the two threads above, dFLMoE (Xie et al., 2025) removes the central server from FL-MoE. Specifcially, clients exchange heads peer-to-peer and fuse them through a locally-trained cross-attention gate, so that no client's knowledge is ever merged into a shared global representation. However, two challenges remain. The framework assumes fully-connected communication, which incurs $\mathcal{O}(N^2)$ per-round cost and does not scale beyond small deployments. The gate of each client is also trained only on local data, so it cannot learn which peer experts are most appropriate for inputs outside its own narrow class subset.

3 METHOD

3.1 PROBLEM FORMULATION

We consider a *decentralized federated learning mixture-of-experts* (DFL-MoE) setting with N clients indexed by $i \in \{1, \ldots, N\}$. We focus on classification, as it provides a widely-adopted benchmark for evaluating expert specialization under non-IID data. Each client i holds a private dataset $\mathcal{D}_i = \{x_i, y_i\}$ drawn from a potentially non-IID distribution, with input x_i and label y_i. Following prior work on shared label distributions (Luo & Wu, 2022), clients may share per-class counts during initialization for grouping. Only these aggregate statistics are disclosed, while raw inputs and sample-level labels remain local. We assume no central server, and clients communicate directly with peers.

The goal is to collaboratively train an MoE model from a distributed expert pool while supporting personalization at each client. The MoE consists of N experts $\{\mathcal{E}_i\}_{i=1}^N$ with parameters $\{\theta_i\}_{i=1}^N$, where each expert $\mathcal{E}_i$ is locally trained on client i's private data $\mathcal{D}_i$ to capture its specialization. Through peer-to-peer communication, each client i obtains an expert subset $\mathcal{H}_i \subseteq \{\mathcal{E}_j\}_{j=1}^N$. Each client i further maintains its own gating network G_i with parameters ϕ_i, which routes inputs to its expert subset. For each input x at client i, the prediction is produced as a gated mixture of expert outputs over $\mathcal{H}_i$.

3.2 OVERVIEW OF DCGMOE

We propose DCGMoE, a DFL-MoE framework that addresses the two challenges above through a *data-aware grouping* mechanism for scalability and a *collaborative gate* for effective expert utilization. Fig. 1 illustrates the overall design.

Architecture. Each client i in DCGMoE maintains the following components. The *body* $F_{b,i}$ extracts client-specific features from raw inputs. The *personalized head* $F_{h,i}$ serves as client i's contribution to the global expert pool. Heads are lightweight, trained locally to capture each client's specialization, and exchanged among clients to form the pool $\mathcal{H}$. Because clients may adopt heterogeneous bodies with incompatible feature spaces, a *feature-space transform* (FST) bridges client i's body output to each expert head. Concretely, FST consists of a set of per-pair projections $\{W_{i,j}\}_{j \in \mathcal{I}_i}$, where $\mathcal{I}_i$ indexes the experts in $\mathcal{H}_i$ (including its own head), and $W_{i,j}$ maps client i's local feature into the input space expected by expert $\mathcal{E}_j$. The *collaborative gate* G_i then assigns per-expert weights to the transformed features and produces the mixture-of-experts prediction. An optional *adapter* placed after the gated aggregation provides a refinement layer, followed by a classifier for the downstream task (see Sec. 4.5).

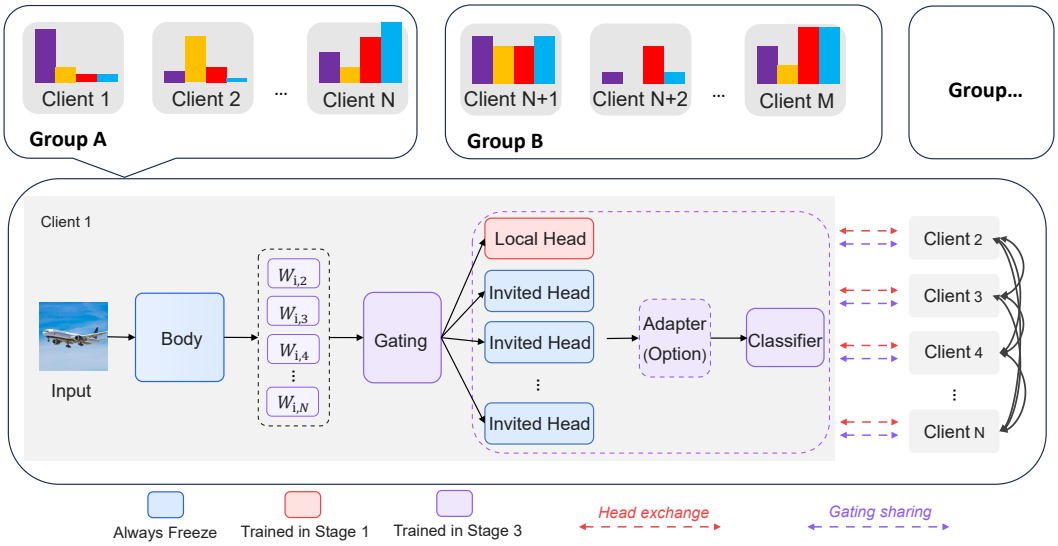


Figure 1: **DCGMoE.** Before training, clients broadcast their label histograms once to compute the grouping. Thereafter, communication stays within groups, and training proceeds independently across groups. **(1)** clients train their heads, and optionally their bodies locally; **(2)** clients exchange heads within their group to form the expert pool; **(3)** clients train the FST, gate, optional adapter, and classifier with the body and heads frozen. The FST remian local and the gate is synchronized.

Algorithm. When the federation size N is large, DCGMoE first performs a one-time *data-aware grouping* step. Each client broadcasts only its label histogram, from which clients are partitioned into K groups, and all subsequent communication occur within each group independently. Training round contains three stages. **Stage 1 (Local Network Training):** each client trains its body (optional) and head on its own data $\mathcal{D}_i$, with the FST and gate frozen, capturing client-specific knowledge. **Stage 2 (Group-wise Expert Exchange):** clients exchange trained heads with their peers, forming the local expert pool. **Stage 3 (Collaborative Gate Training):** each client trains the FST, gate, and the remaining components on its own data with the body and heads frozen. The gate, adapter, and classifier parameters are then synchronized within each group while the FST stays local. We describe data-aware grouping in Sec. 3.3, Stage 1 as well as Stage 2 in Sec. 3.4, and Stage 3 in Sec. 3.5.

3.3 Data-Aware Grouping

To scale to large N, DCGMoE partitions N clients into K disjoint groups so that head exchange and gate synchronization take place exclusively within each group:

$$\{1,\dots,N\} = \bigsqcup_{g=1}^{K} \mathcal{S}_g, \qquad \mathcal{S}_g \cap \mathcal{S}_{g'} = \emptyset \;\; \forall\, g \neq g'. \tag{1}$$

Within each group, each client hosts only $|\mathcal{S}_g|-1 \approx N/K$ peer experts instead of $N-1$. To ensure each group covers a representative slice of the label space, we use a *data-aware* grouping that places *complementary* clients together.

Client complementarity metric. Before training begins, each client broadcasts only its label histogram $\mathbf{c}_i = [n_{i,1},\dots,n_{i,C}] \in \mathbb{N}^C$ (the per-class sample counts). From these histograms we compute a pairwise complementarity score $s(i,j) \in [0,1]$:

$$s(i,j) = \beta \cdot \underbrace{\big(1-\cos(\mathbf{c}_i,\mathbf{c}_j)\big)}_{\text{directional difference}} + (1-\beta)\cdot \underbrace{\frac{H(\mathbf{p}_{i,j})}{\log C}}_{\text{merged balance}}, \tag{2}$$

where $\mathbf{p}_{i,j} = (\mathbf{c}_i+\mathbf{c}_j)/\|\mathbf{c}_i+\mathbf{c}_j\|_1$ is the normalised merged distribution, $H(\cdot)$ is the Shannon entropy, and $\beta \in [0,1]$ trades off the two terms (default $\beta = 0.5$). The first term measures how directionally different the two clients are. This alone is insufficient: two highly-skewed clients can

be maximally dissimilar in direction yet still leave classes uncovered. For instance, $[10, 0, 0]$ and $[0, 0, 10]$ achieve maximal cosine dissimilarity but their merged distribution $[10, 0, 10]$ still has an empty class. The second term, the normalised entropy of the merged distribution, rewards pairings whose joint coverage is closer to uniform across all C classes.

Clustering algorithm. We first construct a similarity matrix using complementary scores $S_{ij} = s(i, j)$. We then learn client embeddings via Multidimensional Scaling (MDS) and partition the clients using balanced k-means clustering. This ensures that clients with high complementary scores are grouped together while enforcing that all K groups are of comparable size.

Communication cost and volume analysis. The communication cost per round is $\mathcal{O}(N^2)$ without grouping and reduces to approximately $\mathcal{O}(N^2/K)$ with grouping under balanced clustering. We now analyze communication volume across the entire federation. Let $|\theta_h|$ and $|\theta_g|$ denote the byte sizes of a head and of the gating with the remaining synchronisable parameters combined, respectively. Under full connection, the per-round volume is

$$T_{\text{full}} = \underbrace{N(N-1) \cdot |\theta_h|}_{\text{head exchange}} + \underbrace{\mathcal{O}(N) \cdot |\theta_g|}_{\text{gate sync}} = \mathcal{O}\big(N^2|\theta_h| + N|\theta_g|\big)\,. \tag{3}$$

With K balanced groups of size $|\mathcal{S}_g| = N/K$, both head exchange and gate synchronization are restricted to within groups and the per-round volume is:

$$T_{DCGMoE} = \sum_{g=1}^{K} \Big[|\mathcal{S}_g|(|\mathcal{S}_g| - 1) \cdot |\theta_h| + \mathcal{O}(|\mathcal{S}_g|) \cdot |\theta_g|\Big] = \mathcal{O}\bigg(\frac{N^2}{K}|\theta_h| + N|\theta_g|\bigg)\,. \tag{4}$$

The head-exchange volume is reduced by approximately a factor of K.

3.4 Local Network Training and Group-wise Expert Exchange

In stage 1, each client i trains its body $F_{b,i}$ and head $F_{h,i}$ on its private data $\mathcal{D}_i$ to capture client-specific knowledge. The FST $\{W_{i,j}\}_{j\in\mathcal{I}_i}$ and the gate G_i are frozen and bypassed. The body output is fed directly into the local head to produce the prediction, that is

$$\hat{y}_i^l = F_{h,i}(F_{b,i}(x_i)). \tag{5}$$

The training objective is the standard task loss:

$$\mathcal{L}_{\text{local},i} = \mathcal{L}(\hat{y}_i^l, y_i), \tag{6}$$

where $\mathcal{L}$ is the task loss (e.g., cross-entropy for classification). Gradient updates always apply to the head, while the FST and gate remain frozen. The body may be trained, or frozen if pretrained.

In Stage 2, each client then sends its head to its peers within the group. After exchange, every client holds a pool of heads to be used as experts in Stage 3. Note that, unlike centralized FL, which sends full model updates to a server every round, DCGMoE shares only the lightweight heads during this exchange stage, while the bodies stay on each client. This keeps the per-round communication cost small and removes the need for a central server.

3.5 Collaborative Gate Training

In this stage, each client uses its expert pool to train the FST, gate, optional adapter, and classifier, while the body and heads stay frozen. For an input x_i, the body output $F_{b,i}(x_i)$ is projected by the per-expert FST into the input space of each expert head, and the gate G_i aggregates the resulting expert outputs into the MoE prediction, that is:

$$\hat{y}_i^m = G_i\Big(\big\{F_{h,j}(W_{i,j}\, F_{b,i}(x_i))\big\}_{j\in\mathcal{I}_i}\Big)\,. \tag{7}$$

Here $\mathcal{I}_i$ contains only expert indices in client i's group. An adapter A_i is optional applied after G_i, acting as an identity mapping when unused and a classifier C_i is applied in the end. The training objective is the standard task loss on the MoE prediction:

$$\mathcal{L}_{\text{gate},i} = \mathcal{L}(\hat{y}_i^m, y_i), \tag{8}$$

with gradients updating the parameters of FST, gate, optional adapter, and classifier.

Each client's gate sees only its local non-IID data, so a gate trained in isolation cannot learn to weight peer experts well. We therefore synchronize the gate and other synchronisable parameters within each group during this stage, while each client's FST $\{W_{i,j}\}_j$ stays local, as it is tied to that client's own body and bodies may be heterogeneous. Let ψ_i represent the set of synchronisable parameters of client i. We study the following two synchronization strategies.

Average Synchronization (Avg-Sync). Clients in each group $\mathcal{S}_g$ update ψ_i in parallel for a few local steps. After each round, the parameters are aggregated based on data size (McMahan et al., 2017):

$$\bar{\psi}_g = \sum_{i \in \mathcal{S}_g} \frac{|\mathcal{D}_i|}{\sum_{k \in \mathcal{S}_g} |\mathcal{D}_k|} \psi_i, \tag{9}$$

where $\bar{\psi}_g$ is broadcast within the group as the initialization for the next round. This aggregation can be performed in a lightweight, decentralized manner to reduce broadcast overhead. The gate, adapter, and classifier are synchronized.

Rolling Synchronization (Roll-Sync). Within each group, a single copy of ψ is passed sequentially from one client to the next: the current holder updates ψ for a few local epochs on its own data and hands the updated ψ to the next client. A full round completes when every client in every group has contributed an update.

Because ψ is overwritten in place, later clients can erase knowledge accumulated from earlier ones. To mitigate this, on receiving ψ the current holder freezes a snapshot ψ^- and augments the task loss with a KL term that anchors ψ to ψ^-:

$$\mathcal{L}_{\text{rolling},i} = \mathcal{L}(\hat{y}_i^m, y_i) + \lambda_{\text{KL}} \cdot \text{KL}\big(p_\psi(\cdot \mid x_i) \parallel p_{\psi^-}(\cdot \mid x_i)\big), \tag{10}$$

where $\lambda_{\text{KL}} \geq 0$ controls the regularization strength.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on three image classification benchmarks with increasing difficulty: Fashion-MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky & Hinton, 2009), and CIFAR-100 (Krizhevsky & Hinton, 2009). For each dataset, we reserve the official test split as a *global test set* shared across all methods. We partition the training split across N clients using Dirichlet sampling (Hsu et al., 2019) with concentration α (default α=0.3). Each client further holds out $1/6$ of its share as a local test subset and trains on the remaining $5/6$.

Architecture. The body is a pretrained ResNet-18 (He et al., 2016), kept frozen throughout our experiments. Each FST is a single linear layer mapping the body output to another expert's input dimension. We use the same body and a simple FST, as they suffice to demonstrate our gating network; heterogeneous body families are validated in prior work and orthogonal to our focus. We denote DCGMoE's architecture as `HxAy`, where x is the number of CNN head blocks and y is the number of adapter blocks. Each block contains two Conv-BN-ReLU-MaxPool units. The default configuration is `H1A1` (one head block, one adapter block), with a lightweight MLP gate. Full experimental details are given in Appendix A.

Baselines. We compare DCGMoE with three groups of baselines: *reference* baselines, *Local* (each client trains only its own body and head; no head exchange and no gate), and *dFLMoE* (Xie et al., 2025); *classical FL* (FedAvg (McMahan et al., 2017), SCAFFOLD (Karimireddy et al., 2020), FedProx (Li et al., 2020)); and *personalised FL* (Ditto (Li et al., 2021), APFL (Deng et al., 2020), FedRep (Collins et al., 2021), LG-FedAvg (Liang et al., 2019)). Classical and personalised FL methods use the PFLlib (Zhang et al., 2025) reference implementations with default hyperparameters and run for 100 communication rounds; DCGMoE runs for 50 rounds.

Metrics. We evaluate accuracy on two test distributions. **Global accuracy** is the mean accuracy of the client models on the shared global test set, which covers the full label distribution. This is our primary metric, reflecting deployment settings in which models receive inputs from the full task distribution. **Local accuracy** is the mean accuracy across clients on their respective held-out test subsets, which follow the same non-IID distributions as their training data.

Table 2: Image classification benchmarks under Dirichlet α=0.3 non-IID partitioning, all with $N = 10$ clients (K=1 no grouping). We report Local accuracy and Global accuracy, together with the corresponding macro-F1 (MF1). Bold indicates the best result in each column.

Method	CIFAR-10				Fashion-MNIST				CIFAR-100			
	Local		Global		Local		Global		Local		Global	
	Acc	MF1	Acc	MF1	Acc	MF1	Acc	MF1	Acc	MF1	Acc	MF1
Local	91.1	75.3	57.3	49.2	92.6	80.2	65.2	58.8	65.1	46.2	26.4	16.8
dFLMoE	**91.5**	**76.5**	59.6	52.0	92.9	81.7	66.8	60.6	**71.1**	**54.2**	33.1	22.4
FedAvg	67.7	39.7	67.6	67.3	89.5	54.9	85.3	84.6	35.8	20.0	34.8	34.2
SCAFFOLD	67.3	39.1	67.1	66.3	90.3	54.5	88.2	88.0	34.2	19.1	33.9	33.0
FedProx	63.7	36.9	64.0	63.6	89.0	54.3	85.4	84.5	31.1	17.5	30.1	29.5
Ditto	67.1	39.5	67.0	66.6	89.2	54.1	85.3	84.7	35.2	20.0	35.0	34.7
APFL	74.2	49.7	67.9	67.5	94.6	78.9	85.7	84.9	34.3	17.2	35.4	34.9
FedRep	80.7	60.6	42.2	34.2	**95.3**	**82.4**	57.7	50.3	36.8	20.3	14.0	10.1
LG-FedAvg	74.7	49.9	33.4	24.9	93.8	73.5	50.4	41.7	34.5	18.2	12.2	8.2
DCGMoE (Ours)	84.8	62.9	**85.0**	**84.2**	88.9	69.9	**88.9**	**88.4**	62.9	43.5	**62.9**	**52.9**

4.2 Comparison on Image Classification

Table 2 shows that DCGMoE achieves the highest global accuracy on all three datasets.

DCGMoE shows larger margins on the more difficult tasks. It exceeds SCAFFOLD on Fashion-MNIST by 0.7 percentage points (88.9 vs. 88.2%) and the strongest baseline, APFL, on CIFAR-10 by 17.1 points (85.0 vs. 67.9%). On CIFAR-100, it reaches 62.9%, while no baseline exceeds 36%; its accuracy is $1.77\times$ that of the next-best baseline.

The baselines show a trade-off between local and global accuracy. *Local* trains each client in isolation and dFLMoE trains its gate only on local data; both fit each client's skewed class distribution but generalise poorly to the global test set, reaching 57.3% and 59.6% global accuracy on CIFAR-10. *Classical FL* methods learn models that generalise through parameter aggregation, but lose client specialisation; their local accuracy on CIFAR-10 reaches at most 67.7%. Among *personalised FL* methods, FedRep achieves the highest local accuracy on Fashion-MNIST (95.3%), whereas FedRep and LG-FedAvg achieve only 12–14% global accuracy on CIFAR-100. This pattern is consistent with *federated overfitting*. Standard deviations of DCGMoE are reported in Appendix A.3. DCGMoE combines competitive local accuracy with the highest global accuracy. Its global accuracy on CIFAR-100 is nearly twice that of the strongest baseline.

4.3 Scaling to Large Federations

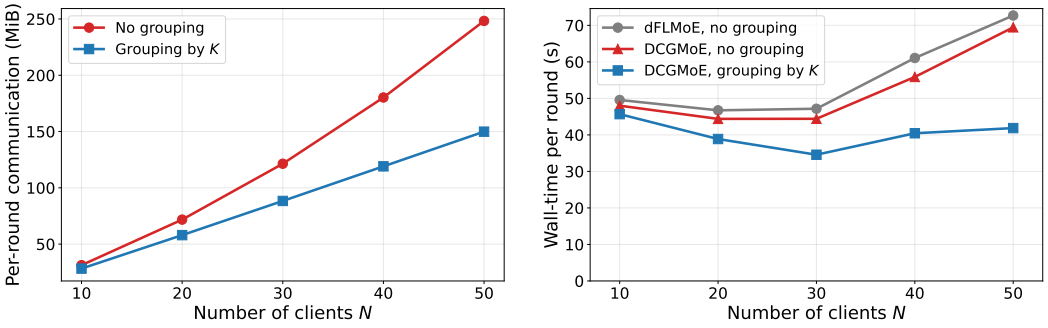
We vary $N \in \{10, 20, 30, 40, 50\}$ on CIFAR-10 and compare two configurations: *All-to-All* (K=1) and *grouping* with $K = \mathcal{O}(\sqrt{N})$ groups. We report dFLMoE only under All-to-All because it has no grouping mechanism. We also include N=100 in Table 3.

Metrics. In addition to global accuracy, we report communication efficiency measures. Per-round communication volume is computed by replacing the asymptotic terms in Equation 4 with constant multiples, e.g., $\mathcal{O}(N) = cN$. We set c=1 for head exchange, c=2 for DCGMoE's Avg-Sync gating, counting one upload and one download per client, and c=0 for dFLMoE's local gating (no exchange). This sums the communication volume over all groups, excluding the one-time histogram exchange before training. Per-round wall time is the mean wall-clock duration of one round.

Per-round communication volume (Fig. 2(a)). Under All-to-All (red), per-round communication grows as $\mathcal{O}(N^2)$ and reaches 248.3 MiB at N=50, while grouping with $K = \mathcal{O}(\sqrt{N})$ (blue) reduces cost to $\mathcal{O}(N^{1.5})$ and reaches 149.9 MiB at N=50. The savings increase with the number of clients.

Per-round wall-time (Fig. 2(b)). Under All-to-All, per-round time grows with N for both dFLMoE (grey) and DCGMoE (red), reaching 73 and 70 s at N=50, since every client trains its gate over all N experts, including its own head. With grouping, each client handles only $s_k \approx \sqrt{N}$ experts and per-round time stays flat at 35–46 s, a $1.66\times$ speed-up at N=50 (69.5→41.9 s).

Accuracy and per-round time at scale (Table 3). DCGMoE maintains higher global accuracy than dFLMoE as the federation grows. dFLMoE's global accuracy falls from 59.6% at N=10 to 52.26%



(a) Per-round communication volume

(b) Per-round wall time

Figure 2: **Scaling behavior on CIFAR-10 (α=0.3).** Grouping uses $K \approx \sqrt{N}$ groups for $N = 10$ to 50. **(a)** Per-round communication of DCGMoE under All-to-All (red) and grouping (blue). **(b)** Mean wall-time per round for dFLMoE (grey) and DCGMoE with and without grouping (red, blue).

Table 3: **Scaling behavior at N=50 and N=100 on CIFAR-10 (α=0.3).** Global test accuracy (%) and average per-round wall-clock time (s). *No group*: All-to-All exchange among all $N-1$ peers. *Group*: clients partitioned into K data-aware groups (K=7 $\approx \sqrt{50}$ at N=50, K=10 $= \sqrt{100}$ at N=100). dFLMoE has no grouping mechanism and is reported only under All-to-All.

Method	50c, no group		50c, group (K=7)		100c, no group		100c, group (K=10)	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time
dFLMoE	52.26	72.7	—	—	42.99	141.6	—	—
DCGMoE (Ours)	**81.69**	**69.5**	**77.97**	**41.9**	**79.10**	**138.9**	**72.92**	**44.5**

at N=50 and 42.99% at N=100, while DCGMoE achieves 81.69% and 79.10% at the latter two scales. The gap therefore widens from about 29 points at N=50 to 36 points at N=100. Grouping reduces per-round time at an accuracy cost. At N=50, it provides a 1.7× speed-up (69.5→41.9 s) with a loss of 3.7 percentage points. At N=100, the speed-up increases to 3.1× (138.9→44.5 s), with a loss of 6.2 points. Grouping thus provides greater speed-ups at larger scales, although the accuracy cost also increases.

4.4 Effect of Grouping Strategy

We examine how partitioning strategy and group size K affect accuracy and communication cost.

Setup. We compare *random grouping* (clients assigned uniformly at random) with our *data-aware grouping* (Sec. 3.3), running DCGMoE on CIFAR-10 with N=50 at $\alpha \in \{0.3, 0.1\}$ and $K \in \{1, 7, 10, 25\}$, where K=7 $\approx \sqrt{N}$ is our default setting. K=1 corresponds to no grouping and provides the reference accuracy.

Results. Accuracy decreases monotonically with K for both strategies (Table 4). At K=25, groups contain only two clients on average, limiting the expert pool and moving the model towards local training. At α=0.3, data-aware grouping improves on random grouping by less than one percentage point at K=7 and K=10. Under stronger heterogeneity (α=0.1), these gains increase to +3.81 and +5.63 points, respectively. These results support grouping clients with complementary data distributions under strong heterogeneity. At our default K=7 $\approx \sqrt{N}$ and α=0.3, grouping costs 3.72 points of accuracy (81.69 → 77.97) while reducing head-exchange traffic by roughly 7×, providing a favourable trade-off.

4.5 Architecture and Gating Ablation

We examine the effects of head depth, adapter depth, and gate synchronisation. We denote architectures as `HxAy` (x head blocks, y adapter blocks) and evaluate six configurations: `H1A0`, `H2A0`, `H3A0`, `H1A1`, `H2A1`, `H1A2`.

Table 4: **Grouping strategy comparison (50 clients).** Data-aware grouping uses cosine complementarity ($\beta = 0.5$) + MDS+k-means. "Δ" = improvement over random.

	Dirichlet $\alpha = 0.3$			Dirichlet $\alpha = 0.1$		
K	Data-aware	Random	Δ	Data-aware	Random	Δ
1	81.69		—	80.92		—
7	**77.97**	77.05	+0.92	**61.69**	57.88	**+3.81**
10	75.34	74.97	+0.37	59.22	53.59	**+5.63**
25	64.76	61.72	+3.04	41.84	38.96	+2.88

Table 5: **Architecture and gating ablation on CIFAR-10 (N=10, α=0.3).** Global test accuracy (%) for 6 architectures × 7 gating strategies. GGT uses pooled training data in both stages. The `H1A1`/Avg-Sync cell is the three-seed mean of Table 2; other cells are single runs. Bold = best realisable (non-GGT) result per architecture.

Model	H+A	OnlyLocal	Local	**Roll-Sync**	**Avg-Sync**	GGT	Batch-R	Group-R
H1A0	1+0	58.3	60.1	**83.8**	83.5	89.9	73.0	34.0
H2A0	2+0	57.4	58.7	**83.2**	79.6	89.5	74.8	23.1
H3A0	3+0	57.2	59.2	82.2	**84.2**	90.0	69.2	27.7
H1A1	1+1	57.3	59.9	66.1	**85.0**	89.8	61.0	50.0
H2A1	2+1	57.9	59.2	66.5	**86.3**	89.5	60.7	44.6
H1A2	1+2	57.3	59.9	65.1	**87.2**	89.5	58.5	49.6

Gating strategies. We evaluate each architecture under seven gating configurations. Our two synchronisation schemes are *Avg-Sync* (the default), which averages gate, adapter, and classifier parameters within each group after each round, and *Roll-Sync*, which passes these shared parameters sequentially across clients and uses KL distillation to regularise the receiver's updates. We also evaluate three variants: *OnlyLocal* disables both head exchange and the gate, so each client trains only its own head (the *Local* baseline of Sec. 4.2); *Batch-Rolling* and *Group-Rolling* apply Roll-Sync at the mini-batch level and within a subgroup, respectively. The remaining two configurations are references. *Local* trains each client's gate on local data without synchronisation, as in dFLMoE. *GGT* is a pooled-data reference in which all clients use the same global training data in both Stage 1 and Stage 3, without gate synchronisation. It relaxes the federated data-sharing constraints.

Results (Table 5). GGT achieves around 90% accuracy across all six architectures. The effect of adapters depends on the synchronisation strategy. Under Avg-Sync, adding an adapter improves accuracy (`H1A0`: $83.5\% \rightarrow$ `H1A1`: 85.0%), with the adapter trained and synchronised together with the gate. Under Roll-Sync, adding an adapter reduces accuracy (`H1A0`: $83.8\% \rightarrow$ `H1A1`: 66.1%). The preferred synchronisation strategy therefore depends on the architecture. Roll-Sync performs better with a shallow head and no adapter (`H1A0`: 83.8% vs. Avg-Sync 83.5%), while Avg-Sync performs better when adapters are present (`H1A1`: 85.0% vs. Roll-Sync 66.1%).

5 Conclusion

We presented DCGMoE addressing two challenges: scaling expert exchange to large client populations, and training gating networks to effectively leverage peer experts under non-IID data. DCGMoE introduces a data-aware grouping mechanism that reduces per-round communication from $\mathcal{O}(N^2)$ to $\mathcal{O}(N^{1.5})$ with $K \approx \sqrt{N}$ balanced groups and a collaborative gate that jointly trains the routing module across clients. Experiments on Fashion-MNIST, CIFAR-10, and CIFAR-100 under non-IID partitioning with α=0.3 show that DCGMoE achieves higher global accuracy than both centralized and decentralized FL baselines, while at scale (N=100) it achieves substantially lower per-round communication and shorter per-round training time than baselines. One limitation is that our grouping strategy currently focuses on label-distribution complementarity. However, the framework is task-agnostic. Other client-side statistics can substitute for label information without modifying the grouping or gating mechanisms, making it extendable to regression, segmentation, and detection. Future work will include more grouping factors such as accuracy, capacity, and latency.

REFERENCES

Vidushi Agarwal, Shruti Mishra, and Sujata Pal. Towards a sustainable blockchain: A peer-to-peer federated learning based approach. *ACM Transactions on Internet Technology*, 24(4), 2024. doi: 10.1145/3680544.

Rahul Atul Bhope, K. R. Jayaram, Praveen Venkateswaran, and Nalini Venkatasubramanian. Shift happens: Mixture of experts based continual adaptation in federated learning. In *Proceedings of the 26th International Middleware Conference*, pp. 455–468. ACM, 2025. doi: 10.1145/3721462.3770784.

Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. FedALT: Federated fine-tuning through adaptive local training with rest-of-world LoRA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 19728–19736, 2026. doi: 10.1609/aaai.v40i24.39054.

Qian Chen, Zilong Wang, Yilin Zhou, Jiawei Chen, Dan Xiao, and Xiaodong Lin. CFL: Cluster federated learning in large-scale peer-to-peer networks. In *Information Security*, pp. 464–472. Springer, 2022. doi: 10.1007/978-3-031-22390-7_27.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 2089–2099, 2021.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

Chen Dun, Mirian Hipolito Garcia, Guoqing Zheng, Ahmed Hassan Awadallah, Robert Sim, Anastasios Kyrillidis, and Dimitrios Dimitriadis. FedJETs: Efficient just-in-time personalization with federated mixture of experts. In *NeurIPS Workshop on Robustness of Few-shot and Zero-shot Learning in Large Foundation Models (R0-FoMo)*, 2023.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Binbin Guo, Yuan Mei, Danyang Xiao, and Weigang Wu. PFL-MoE: Personalized federated learning based on mixture of experts. In *Web and Big Data*, volume 12858 of *Lecture Notes in Computer Science*, pp. 480–486. Springer, 2021. doi: 10.1007/978-3-030-85896-4_37.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

Martin Isaksson, Edvin Listo Zec, Rickard Cöster, Daniel Gillblad, and Sarunas Girdzijauskas. Adaptive expert models for federated learning. In *Trustworthy Federated Learning*, volume 13448 of *Lecture Notes in Computer Science*, pp. 1–16. Springer, 2023. doi: 10.1007/978-3-031-28996-5_1.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Jingang Jiang, Yanzhao Chen, Xiangyang Liu, Haiqi Jiang, and Chenyou Fan. Heterogeneous federated learning with scalable server mixture-of-experts. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 5480–5488, 2025. doi: 10.24963/ijcai.2025/610.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 5132–5143, 2020.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pp. 429–450, 2020.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 6357–6368, 2021.

Zexi Li, Jiaxun Lu, Shuang Luo, Didi Zhu, Yunfeng Shao, Yinchuan Li, Zhimeng Zhang, Yongheng Wang, and Chao Wu. Towards effective clustered federated learning: A peer-to-peer framework with adaptive neighbor matching. *IEEE Transactions on Big Data*, 10(6):812–826, 2024. doi: 10.1109/TBDATA.2022.3222971.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

Edvin Listo Zec, Olof Mogren, John Martinsson, Leon René Sütfeld, and Daniel Gillblad. Specialized federated learning using a mixture of experts. *arXiv preprint arXiv:2010.02056*, 2020.

Fan Liu, Bikang Pan, Zhongyi Wang, Xi Yao, Xiaoying Tang, Jingya Wang, and Ye Shi. FLEx: Personalized federated learning for mixture-of-experts LLMs via expert grafting. *arXiv preprint arXiv:2506.00965*, 2025. doi: 10.48550/arXiv.2506.00965.

Yuan Liu, Zhengpeng Ai, Shuai Sun, Shuangfeng Zhang, Zelei Liu, and Han Yu. Fedcoin: A peer-to-peer payment system for federated learning. In *Federated Learning: Privacy and Incentive*, pp. 125–138. Springer, 2020.

Jun Luo and Shandong Wu. FedSLD: Federated learning with shared label distribution for medical image classification. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5, 2022. doi: 10.1109/ISBI52829.2022.9761404.

Jun Luo, Chen Chen, and Shandong Wu. Mixture of experts made personalized: Federated prompt learning for vision-language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

Hanzi Mei, Dongqi Cai, Ao Zhou, Shangguang Wang, and Mengwei Xu. FedMoE: Personalized federated learning via heterogeneous mixture of experts. *arXiv preprint arXiv:2408.11304*, 2024.

Saeedeh Parsaeefard, Sayed Ehsan Etesami, and Alberto Leon Garcia. Robust federated learning by mixture of experts. *arXiv preprint arXiv:2104.11700*, 2021.

Jiaxing Qi, Zhongzhi Luan, Shaohan Huang, Carol Fung, Hailong Yang, and Depei Qian. FDLoRA: Personalized federated learning of large language model via dual LoRA tuning. *arXiv preprint arXiv:2406.07925*, 2024.

Zhen Qin, Xueqiang Yan, Mengchu Zhou, and Shuiguang Deng. BlockDFL: A blockchain-based fully decentralized peer-to-peer federated learning framework. In *Proceedings of the ACM Web Conference 2024*, pp. 2914–2925. ACM, 2024. doi: 10.1145/3589334.3645425.

Matthias Reisser, Christos Louizos, Efstratios Gavves, and Max Welling. Federated mixture of experts. *arXiv preprint arXiv:2107.06724*, 2021.

Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. BrainTorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

Jose L. Salmeron, Irina Arévalo, and Antonio Ruiz-Celma. Benchmarking federated strategies in peer-to-peer federated learning for biomedical data. *Heliyon*, 9(6):e16925, 2023. doi: 10.1016/j.heliyon.2023.e16925.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Han Wang, Luis Muñoz-González, David Eklund, and Shahid Raza. Non-IID data re-balancing at IoT edge with peer-to-peer federated learning for anomaly detection. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 153–163. ACM, 2021. doi: 10.1145/3448300.3467827.

Lei Wang, Jieming Bian, Letian Zhang, and Jie Xu. Adaptive LoRA experts allocation and selection for federated fine-tuning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2553.

Panlong Wu, Kangshuo Li, Ting Wang, and Fangxin Wang. FedMS: Federated learning with mixture of sparsely activated foundations models. *arXiv preprint arXiv:2312.15926*, 2023. doi: 10.48550/arXiv.2312.15926.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL `http://arxiv.org/abs/1708.07747`.

Luyuan Xie, Tianyu Luan, Wenyuan Cai, Guochen Yan, Zhaoyu Chen, Nan Xi, Yuejian Fang, Qingni Shen, Zhonghai Wu, and Junsong Yuan. dFLMoE: Decentralized federated learning via mixture of experts for medical data analysis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10203–10213, 2025. doi: 10.1109/CVPR52734.2025.00954.

Liping Yi, Han Yu, Gang Wang, Xiong Liu, and Qinghua Hu. pFedMoE: Data-level personalization with mixture of experts in model-heterogeneous personalized federated learning. *IEEE Transactions on Knowledge and Data Engineering*, 38(3):1905–1918, 2026. doi: 10.1109/TKDE.2026.3656194.

Ziwei Zhan, Wenkuan Zhao, Yuanqing Li, Weijie Liu, Xiaoxi Zhang, Chee Wei Tan, Chuan Wu, Deke Guo, and Xu Chen. FedMoE-DA: Federated mixture of experts via domain aware fine-grained aggregation. In *2024 20th International Conference on Mobility, Sensing and Networking (MSN)*, pp. 122–129. IEEE, 2024. doi: 10.1109/MSN63567.2024.00027.

Boyang Zhang, Xiaobing Chen, Songyang Zhang, Shuai Zhang, Xiangwei Zhou, Jian Zhang, and Mingxuan Sun. FLEX-MoE: Federated mixture-of-experts with load-balanced expert assignment for edge computing. *arXiv preprint arXiv:2512.23070*, 2026a.

Jianqing Zhang, Yang Liu, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Jian Cao. PFLlib: A beginner-friendly and comprehensive personalized federated learning library and benchmark. *Journal of Machine Learning Research*, 26(50):1–10, 2025.

Yicheng Zhang, Zhen Qin, Zhaomin Wu, Jian Hou, and Shuiguang Deng. Personalized federated fine-tuning for llms via data-driven heterogeneous model architectures. In *Proceedings of the ACM Web Conference 2026*, WWW '26, pp. 5099–5110, 2026b.

Yang Zhao, Jun Zhao, Linshan Jiang, Rui Tan, Dusit Niyato, Zengxiang Li, Lingjuan Lyu, and Yingbo Liu. Privacy-preserving blockchain-based federated learning for IoT devices. *IEEE Internet of Things Journal*, 8(3):1817–1829, 2021. doi: 10.1109/JIOT.2020.3017377.

A Experimental Details

A.1 Training and Architecture

Table 6 lists settings for DCGMoE and dFLMoE. Avg-Sync averages the gate, adapter and classifier within each group once per round, weighted by local dataset size; the FST remains local. Code will be released upon acceptance.

Table 6: Training settings for DCGMoE and dFLMoE.

Setting	Value
Communication rounds	50
Local epochs per round	2 in Stage 1; 2 in Stage 3
Optimiser	SGD; momentum 0.9; weight decay 5×10^{-4}
Learning rate	0.01 in Stage 1; 0.005 in Stage 3; both halved every 10 rounds
Batch size / gradient clipping	128 / gradient norm capped at 5
Random seed	42 (single run)
Stage 1 loss	$0.5\,\mathrm{CE}(\text{local head}) + 0.5\,\mathrm{CE}(\text{MoE})$; the MoE term is active from round 2, once invited experts are present
Stage 1 updates	Head only; body, FST, gate and adapter frozen
Stage 3 updates	FST, gate, adapter and classifier; body and all heads frozen

Table 7 describes `H1A1`. dFLMoE uses the same body and head, no adapter, and an unsynchronised cross-attention gate (body queries, expert-output keys and values, temperature 1.0).

Table 7: Default `H1A1` architecture. E is the number of experts in the client's pool, and C is the number of classes.

Component	Specification
Body	ResNet-18, ImageNet-pretrained and frozen; 512-d output
Head (`H1`)	One CNN block: two Conv(32)–BN–ReLU layers followed by MaxPool; 512-d output
Adapter (`A1`)	One CNN block: two Conv(128)–BN–ReLU layers followed by MaxPool
Gate	MLP $512 \to 256 \to 128 \to E$; LayerNorm and dropout 0.1
FST	One 512×512 linear map per (client, expert) pair; identity-initialised and client-local
Classifier	Linear layer to C classes

A.2 Data and Grouping

Training data are split across N clients by Dirichlet sampling ($\alpha = 0.3$; also 0.1 in Table 4), with $1/6$ held out per client for local testing. Global testing uses the official test split.

CIFAR-10 and CIFAR-100 use 32×32 RGB inputs, normalised by their respective channel means and standard deviations. Training uses 32×32 crops (reflect padding 4), horizontal flips, rotations of $\pm 7°$, affine transforms (shear $10°$, scale 0.8–1.2), and colour jitter of 0.2.

Fashion-MNIST images are resized from 28×28 to 32×32 using bicubic interpolation, replicated to three channels, and normalised with mean 0.5 and standard deviation 0.5. No augmentation is applied.

We compute the complementarity score in Eq. (2) with $\beta = 0.5$, apply MDS to obtain two-dimensional embeddings, and run k-means followed by rebalancing to equal group sizes. The partition is computed once from the shared label histograms before training and remains fixed throughout, using seed 42. We set $K = \mathrm{round}(\sqrt{N})$: $K = 3, 4, 5, 6, 7$ for $N = 10, 20, 30, 40, 50$, respectively, and $K = 10$ for $N = 100$.

A.3 VARIANCE ACROSS SEEDS

Table 8 reports DCGMoE in the setting of Table 2 as the mean and standard deviation over three seeds (42, 43, 44); the seed controls both the Dirichlet partition and the initialisation.

Table 8: DCGMoE (N=10, α=0.3), mean $\pm$ standard deviation over three seeds.

	Local		Global	
Dataset	Acc	MF1	Acc	MF1
CIFAR-10	84.8 ± 1.1	62.9 ± 2.2	85.0 ± 0.6	84.2 ± 0.6
Fashion-MNIST	88.9 ± 0.0	69.9 ± 1.2	88.9 ± 0.1	88.4 ± 0.2
CIFAR-100	62.9 ± 0.2	43.5 ± 0.3	62.9 ± 0.1	52.9 ± 0.3