FEDDAG: CLUSTERED FEDERATED LEARNING VIA GLOBAL DATA AND GRADIENT INTEGRATION FOR HETEROGENEOUS ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) enables a group of clients to collaboratively train a model without sharing individual data, but its performance drops when client data are heterogeneous. Clustered FL tackles this by grouping similar clients. However, existing clustered FL approaches rely solely on either data similarity or gradient similarity; however, this results in an incomplete assessment of client similarities. Prior clustered FL approaches also restrict knowledge and representation sharing to clients within the same cluster. This prevents cluster models from benefiting from the diverse client population across clusters. To address these limitations, FEDDAG introduces a clustered FL framework, FEDDAG, that employs a weighted, class-wise similarity metric that integrates both data and gradient information, providing a more holistic measure of similarity during clustering. In addition, FEDDAG adopts a dual-encoder architecture for cluster models, comprising a primary encoder trained on its own clients' data and a secondary encoder refined using gradients from complementary clusters. This enables cross-cluster feature transfer while preserving cluster-specific specialization. Experiments on diverse benchmarks and data heterogeneity settings show that FEDDAG consistently outperforms state-of-the-art clustered FL baselines in accuracy.

1 INTRODUCTION

Federated Learning (FL) enables users/clients to collaboratively train a model on their data without sharing it with other clients or a central entity (McMahan et al., 2017). However, diversity in user behavior results in heterogeneous data distributions, known as *non-identically independently distributed* (non-IID) data, across clients. This heterogeneity can lead to slower convergence and suboptimal accuracy of the global model (Kairouz et al., 2021). More specifically, non-IID data can arise due to various factors, including class/label skew, feature skew, quantity shift, concept shift, and concept drift — common types of data heterogeneity. *Class/label skew* refers to the non-identical distribution of labels/classes at different clients, e.g., the absence of a label at one client while the same label is present at other clients (Zhang et al., 2022). *Feature skew* occurs when distributions vary due to different personalization nuances, e.g., an alphabet letter can be written in different ways (Li et al., 2021). *Quantity shift* happens when different clients have different amounts of data (Wang et al., 2021), e.g., an online retailer with millions of transaction records is compared to a local store with only a few hundred records. *Concept shift* happens when different clients assign the same label to fundamentally different data samples due to variations in local data distributions or labeling criteria (Kang et al., 2024).

Clustered FL handles non-IID data effectively, especially when distinct groups of clients display substantial variations in their data distributions (Ghosh et al., 2020; Guo et al., 2024; Vahidian et al., 2023). In clustered FL, clients are grouped into clusters based on their similarities in their data distributions, and each cluster trains its own model tailored to its specific data. However, despite their advantages, existing clustered FL approaches suffer from the following limitations:

1. Improper Similarity Method. Cluster FL approaches use either data or gradient alone to compute similarity for clustering. Cluster FL approaches (Sattler et al., 2020; Long et al., 2023; Ghosh et al., 2020) that use gradients or loss values to cluster clients can group clients incorrectly due to the

high dimensionality of data or the presence of various skews in client data (Vahidian et al., 2023). Other drawbacks of these approaches include: requiring each client to evaluate multiple global models every round (Ghosh et al., 2020; Licciardi et al., 2025), delaying cluster formation until many training iterations, and requiring clients to upload full model updates (Sattler et al., 2020).

On the other hand, the data-based approach, such as PACFL (Vahidian et al., 2023), only considers label skew and does not account for skew issues like concept shift. Moreover, PACFL defines inter-client similarity as the minimum cosine angle between the clients' feature subspaces. However, by relying on the smallest angle across the subspaces, PACFL may yield high similarity even when only a small portion of the clients' data is similar, while the remaining subspaces are vastly dissimilar.

2. Global Representation Sharing. Existing Clustered FL approaches restrict knowledge sharing to clients within the same cluster. This prohibits clients across clusters to benefit from low-level latent representations. One way FedSoft (Ruan & Joe-Wong, 2022) and FedRC (Guo et al., 2024) address this issue by incorporating multiple cluster models through soft clustering with learnable cluster importance weights. However, in these approaches, a client's model becomes a noisy blend of several cluster models. While this blending may occasionally benefit data that aligns with several clusters, the added noise from unrelated clusters may degrade the performance on the client's primary dataset, since the model is no longer explicitly optimized for its own data.

3. Limited Consideration of Distribution Skews. Clustered FL techniques (Sattler et al., 2020; Ghosh et al., 2020; Vahidian et al., 2023; Licciardi et al., 2025) primarily address label skew. However, these approaches do not account for concept shift or quantity shift.

4. Predefined Cluster Numbers. Existing clustered FL approaches lack adaptive mechanisms for automatically adjusting the number of clusters. For example, IFCA (Ghosh et al., 2020) requires the optimal number of clusters to be specified in advance. Sattler et al. (2020) adopts a recursive strategy to split clusters when gradients converge to a stationary point but cannot merge clusters when needed, such as upon the arrival of new clients. Zeng et al. (2023) supports merging clusters but not splitting them. Li et al. (2024) evaluates candidate clustering using traditional clustering metrics that do not account for the unique characteristics of FL setting.

These limitations raise the following crucial question:

How can we overcome the above challenges posed by various skews in heterogeneous data distributions by utilizing both data and gradient information to dynamically cluster clients and enabling representation sharing among clusters in FL?

Our contribution. This work proposes a novel algorithm, entitled clustered Federated Learning via global DatA and Gradient integration (FEDDAG). FEDDAG introduces a novel method to compute similarities among clients and an innovative approach that combines data and gradient information for improved client grouping. To combine data- and gradient-based similarity to achieve a more accurate similarity matrix, FEDDAG assigns each client a weight that indicates how much emphasis to place on data versus gradient information. FEDDAG optimizes these weights using an entropy-based loss that sharpens the final adjacency matrix. To further improve client similarity estimation, FEDDAG extends the data-based approach PACFL (Vahidian et al., 2023) by performing class-wise comparisons rather than comparing entire data subspaces—restricting comparisons to subspaces corresponding to the same class across clients. This approach yields a more accurate similarity metric and naturally accounts for concept shift. In addition, FEDDAG assigns weights to the class-wise similarity values to address quantity shift. FEDDAG also improves upon the existing gradient-based similarity so that client computes gradients for at most one model per round and transmits only a compressed gradient.

These above mechanisms improve similarity computation and lead to better client clustering. We further enhance FEDDAG by employing a dual-encoder architecture to enable effective representation sharing across clusters. During the training phase, each cluster model consists of: (i) a primary encoder, optimized using the cluster's own client data, and (ii) a secondary encoder, designed to learn complementary features from other clusters. The outputs of the two encoders are concatenated along the feature dimension, and a classifier is trained on the combined representation. This design facilitates cross-cluster knowledge transfer while preserving cluster-specific specialization.

Compared to prior works, to our knowledge, FEDDAG is the only work that addresses all four types of data heterogeneity: label skew, feature skew, concept shift, and quantity shift. FEDDAG accounts

for concept shift by performing class-wise comparisons when computing similarity between clients' data. Additionally, FEDDAG introduces an adaptive clustering mechanism that automatically determines the optimal number of clusters through a novel evaluation metric. Specifically, it generates a range of candidate clusterings using hierarchical clustering (HC) (Day & Edelsbrunner, 1984) and evaluates them with a novel federated-aware metric that rewards compact cluster formation while penalizing over-splitting.[1] In summary, the contributions of this paper are as follows:

1. A new clustered FL algorithm, FEDDAG, that combines both data and gradient similarity for better client clustering and further improves data similarity estimation with a class-wise weighted method.
2. FEDDAG introduces a novel method for knowledge and representation sharing across clusters by employing a dual-encoder architecture.
3. This work introduces a novel federated-aware metric to evaluate candidate clusterings and automatically determine the optimal number of clusters.
4. We evaluate FEDDAG under non-IID data, having class skew, feature skew, concept shift, and quantity shift, and across different degrees of heterogeneity (e.g., high vs. low). Table 1 reports the accuracy of FEDDAG in comparison to existing clustered FL methods. Detailed experimental results are provided in §5.

2 LITERATURE REVIEW

Clustered FL techniques address distribution shift by grouping clients based on their data distributions. PACFL (Vahidian et al., 2023) clusters clients by analyzing principal angles between client data subspaces, but it ignores label information, making it prone to incorrect clustering under concept shift. Another line of work (Ghosh et al., 2020; Licciardi et al., 2025) uses loss values on gradients to iteratively cluster clients each training round. Other methods group clients via gradient similarity (Duan et al., 2021a; Sattler et al., 2020), while soft clustering enables clients to join multiple clusters (Ruan & Joe-Wong, 2022; Guo et al., 2024). A recent approach Zhang et al. (2024) develops adaptive clustering based on cosine similarity between dimensionally-reduced models. Additional methods, such as Long et al. (2023); Marfoq et al. (2021); Wu et al. (2023), rely on maximizing log-likelihood functions or modeling joint distributions. Compared to these, FEDDAG combines data and gradient information for better clustering and enables cross-cluster knowledge transfer while preserving cluster-specific specialization.

Table 1: Accuracy (%) of FEDDAG vs. clustering baselines under non-IID label skew (20%) and quantity shift (Dirichlet α'=1).

Algorithm	Technique	CIFAR-10	FMNIST
PACFL	Data (D)	90.45±0.30	94.41±0.31
CFL	Gradient (G)	72.80±0.66	86.97±0.23
IFCA	Gradient (G)	89.68±0.17	94.03±0.09
FEDDAG (Ours)	**D + G + Global Feature Sharing**	**94.53±0.12**	**96.82±0.18**

3 FEDDAG ALGORITHM

FEDDAG, a framework for clustered FL, can be formulated as an empirical risk minimization (ERM) problem over N clients, each holding a local dataset $D_i=(X_i, Y_i)$, where X_i and Y_i denote the input samples and labels, respectively. The data can be non-iid and may exhibit various skews (as discussed in §1). The server partitions the clients into Z clusters $\mathbb{C}_1, \ldots, \mathbb{C}_Z$. The objective is to minimize the local loss $\mathcal{L}(Y_i, F_{z(i)}(X_i))$ for each client $i \in N$, where $z(i)$ is the cluster assignment determined by FEDDAG. Simplified FEDDAG cluster-level model is defined as:

$$F_z(\cdot) = \psi\big(\phi(\cdot; \Theta_z^f);\ \Theta_z^c\big) \tag{1}$$

Here, ϕ is the feature encoder and ψ is the classifier head. FEDDAG also supports a more expressive dual-encoder architecture, where the outputs of two encoders are jointly processed by the classifier head, as represented below:

$$F_z(\cdot) = \psi\big(\phi^{(1)}(\cdot; \Theta_z^{1f}), \phi^{(2)}(\cdot; \Theta_z^{2f})\ ;\ \Theta_z^c\big) \tag{2}$$

[1] Over-splitting is a common issue in HC for FL that can violate key principles of FL by producing degenerate clusters with very few clients (Licciardi et al., 2025).

We describe FEDDAG (see Algorithm 1) in two parts. First, we introduce the weighted class-wise approach (Algorithm 2 in Appendix A.3) for computing data similarity among clients and combine both data and gradient information to improve clustering (Algorithm 3 in Appendix A.3). The improved clustering can be directly used for traditional clustered FL, resulting in higher accuracy (see §5). We then further enhance FEDDAG with a dual-encoder mechanism (described in §4) that enables inter-cluster representation sharing during FL training, which further increases FEDDAG's performance. An illustration of FEDDAG is shown in Figure 2 in Appendix A.3, and its components are described below.

3.1 GRADIENT-BASED SIMILARITY

High-level idea. FEDDAG introduces a lightweight method for computing gradient similarity. Prior approaches such as Sattler et al. (2020) and Kim et al. (2024b) periodically send gradient updates to the server to measure client similarity. In contrast, our approach has each client first train locally on its own data (without federation) for a few rounds to partially converge the gradients. We observed that two such rounds (10 local steps each) are sufficient to achieve partial convergence, making inter-client similarity more distinguishable (see experiments on Local Steps (t_g) in Appendix §B.2). To further reduce communication, FEDDAG transmits a k-sparse version of the gradients (retaining only k coordinates) to the server for similarity computation (Wangni et al., 2018).

Details of the method. Each client $i \in N$ is initialized with random parameters θ_i^0 and performs local training (without federation) on D_i for $t_g = 2$ rounds (see Appendix §B.2) to obtain a gradient update Δ^i. The update is k-sparsified—retaining only a small random subset of entries (typically 1–2%) (Wangni et al., 2018). The sparsified update $\tilde{\Delta}^i$ is then sent to the server, which constructs a pairwise similarity matrix. The similarity $\mathcal{G}_{i,j}$ between clients i and j is computed as:

$$\mathcal{G}_{i,j} = \cos^{-1}\left(\frac{\langle \tilde{\Delta}^i, \tilde{\Delta}^j \rangle}{\|\tilde{\Delta}^i\| \, \|\tilde{\Delta}^j\|}\right) \times \frac{180}{\pi}, \quad \forall i, j \in N. \tag{3}$$

3.2 WEIGHTED CLASS-WISE DATA-BASED SIMILARITY

High-level idea. Our goal is to construct a data-based similarity matrix that will be fused with the gradient matrix for clustering. Unlike the existing data-based approach, PACFL (Vahidian et al., 2023), which compares the entire data subspaces of two clients, we measure similarity in a class-wise manner and assign weights to the class-level similarities to compute the final client similarity.

Details of the method. Let C be the total number of classes, and $D_{i,c}$ the data of client $i \in N$ for class $c \in C$. Each client applies truncated SVD (Klema & Laub, 1980) on the transpose of $D_{i,c}$ to compute p principal vectors per class, denoted $U_c^i = [u_1, \ldots, u_p]$. These vectors are then sent to the server to compute the data similarity matrix.[2] For each class c, the server computes the principal angle (Jain et al., 2013) between U_c^i and U_c^j, indicating the similarity between clients i and j as:

$$\mathcal{V'}_{i,j,c} = \min_{\mathbf{v} \in U_c^i, \mathbf{x} \in U_c^j} \cos^{-1}\left(\frac{|\mathbf{v}^\top \mathbf{x}|}{\|\mathbf{v}\| \cdot \|\mathbf{x}\|}\right), \quad \forall i, j \in N. \tag{4}$$

If class c is present in only one of the clients, $\mathcal{V'}_{i,j,c} = 90°$; if in neither, $\mathcal{V'}_{i,j,c} = 0°$. Next, the server assigns weights $\mathcal{W}_{i,j,c}$ to each class-wise similarity $\mathcal{V'}_{i,j,c}$ to reflect class frequency differences (i.e., quantity skew) between clients i and j. This weighting scheme ensures that larger differences in class frequency lead to higher dissimilarity values. The weights are computed as:

$$\mathcal{W}_{i,j,c} = \frac{\max(\ln(|D_{i,c}| + \epsilon), \ln(|D_{j,c}| + \epsilon))}{\min(\ln(|D_{i,c}| + \epsilon), \ln(|D_{j,c}| + \epsilon))} \tag{5}$$

then min–max normalized to a bounded range $[1-\delta, 1+\delta]$, where $\delta > 0$ controls the server's tolerance to frequency imbalance. The final similarity between clients i and j is:

$$\mathcal{V}_{i,j} = \tfrac{1}{|C|} \sum_{c=1}^{C} \mathcal{V'}_{i,j,c} \, \mathcal{W'}_{i,j,c}, \qquad \mathcal{W'}_{i,j,c} \leftarrow \text{normalized } \mathcal{W}_{i,j,c}. \tag{6}$$

[2] In FEDDAG, clients share a small set of principal vectors and class frequency information with the server to compute similarity. These principal vectors are not actual client data, but a linear combination of them. Moreover, the number of principal vectors shared with the server is less than 1% of the size of the dataset for each class per client. This approach aligns with prior works, such as PACFL(Vahidian et al., 2023).

Algorithm 1: FEDDAG Algorithm

Input: Number of clients N, sampling rate $R \in (0, 1]$, C classes
Output: Updated global model parameters

Initialize client $i \in N$ with random θ_i^0
for *each round* $t = 0, 1, \ldots$ **do**
  $m \leftarrow \max(R \cdot N, 1)$ `// Sampling rate`
  $S_m \leftarrow \{i_1, ..., i_m\}$ `// Set of` m `sampled clients`
  for *each client* $i \in N$ ***in parallel*** **do**
    if $t \leq t_g$ **then**
      Local training of θ_i^0 with client i local data (no federation)
      if $t = t_g$ **then**
        Client i sends sparsified local model update $\tilde{\Delta}^i$ to server
        Client i performs SVD and extracts principal vectors $U_c^i,\ \forall c \in C$ and sends to server
        Server forms $\mathcal{A} \leftarrow$ `ProximityMatrix(`$U^*, \tilde{\Delta}^*$`)` (Algorithm 2)`// Adjacency matrix`
        Server computes optimal Clustering $\{\mathbb{C}_1, \ldots, \mathbb{C}_Z\} \leftarrow$ `OptimalClustering(`$\mathcal{A}, S_\alpha$`)` (Algorithm 3)`// Find best clustering`
        Server computes the *CC-Graph* H as per Eq. 12
        Server initiates Θ_z^{1f} as in Eq. 20, and Θ_z^{2f} and Θ_z^c randomly `// cluster encoder initialization`
    else
      Server sends $\{\Theta_{z(i)}^{1f}, \Theta_{z(i)}^{2f}, \Theta_{z(i)}^c\}$ and $\Theta_{z(i)}^{2f'} = \sum_{j:H(j,z(i))=1} \Theta_j^{2f}$ to client i
      Client i sets $(\theta_i^{1f}, \theta_i^c) \leftarrow (\Theta_{z(i)}^{1f}, \Theta_{z(i)}^c)$ and trains them via SGD as in Eq. 15 `// primary training phase`
      Client i sets $\theta_i^{2f'} \leftarrow \Theta_{z(i)}^{2f'}$ and updates via SGD as in Eq. 18 `// Secondary training phase`
      Client i broadcasts $(\theta_i^{1f}, \theta_i^c)$ and $\theta_i^{2f'}$ to server
  if $t \geq t_g$ **then**
    `// Executed after clusters are formed` $t \geq t_g$
    for *each cluster* $z = 1$ *to* Z **do**
      Update Θ_z^{1f} and Θ_z^c, as in Eq. 16
      Update learner cluster $\Theta_{j:H(j,z)=1}^{2f}$, as in Eq. 19

3.3 COMBINING DATA & GRADIENT — ALGORITHM 2

High-level idea. After constructing the data and gradient similarity matrices, FEDDAG applies min-max normalization and then combines them into a single proximity matrix, which serves as the adjacency matrix for clustering.

Details of the method. Given the normalized $\hat{\mathcal{V}}_{i,j}$ and $\hat{\mathcal{G}}_{i,j}$, FEDDAG learns a weight vector $\mathbf{w} = (w_1, \ldots, w_N)^\top \in [0, 1]^N$, where each w_i is assigned to client i to control the relative importance of gradient versus data similarity. FEDDAG then fuses the normalized matrices to construct the proximity matrix as follows:

$$\mathcal{A}_{i,j} = w_i \hat{\mathcal{G}}_{i,j} + \big(1 - w_i\big)\hat{\mathcal{V}}_{i,j}, \quad 1 \leq i < j \leq N, \quad \mathcal{A}_{j,i} = \mathcal{A}_{i,j}. \tag{7}$$

FEDDAG optimizes $\mathbf{w}$ by minimizing the entropy loss:

$$\mathcal{L}_{\text{en}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{\mathcal{A}}_{i,j} \log \tilde{\mathcal{A}}_{i,j},\ \tilde{\mathcal{A}}_{i,j} = \frac{e^{\mathcal{A}_{i,j}}}{\sum_{k=1}^{N} e^{\mathcal{A}_{i,k}}} \tag{8}$$

where $\tilde{\mathcal{A}}_{i,j}$ is the row-wise softmax normalization of $\mathcal{A}_{i,j}$. In Eq. 8, the loss $\mathcal{L}_{\text{en}}$ sharpens each row of the fused matrix $\mathcal{A}_{i,j}$, encouraging each client to retain only its strongest neighbors (Ghasedi Dizaji et al., 2017). This, in turn, guides $\mathbf{w}$ to favor the view (i.e., data or gradient) that leads to a more clusterable affinity structure. FEDDAG learns the weight vector $\mathbf{w}$ using a lightweight multi-layer perceptron (MLP) (Almeida, 2020) trained via gradient descent to minimize the entropy loss $\mathcal{L}_{\text{en}}$. Finally, FEDDAG constructs the proximity matrix using the learned $\mathbf{w}$ as shown in Eq. 7.

3.4 OPTIMAL CLUSTERING — ALGORITHM 3

High-level idea. FEDDAG introduces an adaptive clustering mechanism that automatically identifies the optimal number of clusters. This mechanism incorporates a novel federated-aware metric to evaluate clustering quality.

Details of the method. Given the proximity matrix $\mathcal{A}_{i,j}$, the server applies agglomerative hierarchical clustering (HC). In HC, the *clustering threshold* $\alpha \in (0, 1]$ controls merges: clusters with pairwise distances below α are merged. Smaller α yields more clusters; larger α merges more broadly. The server iterates over different α values to generate candidate clusterings $\{\mathbb{C}_1, \ldots, \mathbb{C}_Z\}$, each with a distinct number of clusters Z. Each clustering is evaluated using two metrics. Compact-

Table 2: Exp 1: Performance comparison for Data Distribution I with a high degree of quantity shift (Dirichlet $\alpha' = 0.25$)

	20% Label Skew				30% Label Skew			
Algorithm	**CIFAR-10**	**FMNIST**	**SVHN**	**CIFAR-100**	**CIFAR-10**	**FMNIST**	**SVHN**	**CIFAR-100**
FedAvg	42.02 ± 1.17	53.11 ± 0.31	69.79 ± 0.51	47.16 ± 0.91	54.24 ± 0.08	72.86 ± 0.40	64.15 ± 0.64	50.99 ± 1.35
FedProx	43.98 ± 0.17	53.61 ± 0.20	74.75 ± 0.27	50.56 ± 0.70	54.99 ± 0.20	68.22 ± 0.16	64.80 ± 0.25	48.66 ± 0.80
PerFedAvg	81.09 ± 0.35	86.51 ± 0.19	89.20 ± 0.05	65.59 ± 0.02	77.45 ± 0.24	89.77 ± 0.15	88.23 ± 0.31	57.38 ± 0.10
FedSoft	76.44 ± 0.18	84.58 ± 0.14	83.75 ± 0.33	62.54 ± 0.41	72.48 ± 0.17	85.15 ± 0.17	82.43 ± 0.40	55.24 ± 0.43
PACFL	86.93 ± 0.40	91.90 ± 0.47	89.88 ± 0.25	66.11 ± 0.29	84.66 ± 0.29	91.96 ± 0.25	90.48 ± 0.23	58.30 ± 0.56
CFL	68.67 ± 0.76	81.90 ± 0.10	79.83 ± 0.38	57.38 ± 0.95	67.57 ± 0.69	80.64 ± 0.21	75.21 ± 0.09	49.63 ± 1.29
CFL-GP	85.25 ± 0.17	89.13 ± 0.35	87.83 ± 0.22	67.89 ± 0.20	83.98 ± 0.28	91.14 ± 0.14	90.01 ± 0.11	59.71 ± 0.76
FedGWC	85.97 ± 0.13	91.02 ± 0.17	89.35 ± 0.10	69.19 ± 0.48	83.58 ± 0.21	91.45 ± 0.12	88.94 ± 0.15	56.52 ± 0.40
FedRC	75.12 ± 0.28	88.32 ± 0.23	88.05 ± 0.30	63.25 ± 0.37	76.48 ± 0.37	88.12 ± 0.25	85.78 ± 0.38	54.32 ± 0.33
IFCA	86.64 ± 0.13	90.93 ± 0.17	89.51 ± 0.10	69.08 ± 0.48	83.45 ± 0.37	91.50 ± 0.11	88.81 ± 0.09	56.33 ± 0.40
FEDDAG*	88.67 ± 0.18	92.75 ± 0.22	91.87 ± 0.26	70.37 ± 0.33	86.95 ± 0.21	92.18 ± 0.15	90.97 ± 0.13	60.84 ± 0.65
FEDDAG	$\mathbf{90.76 \pm 0.12}$	$\mathbf{93.82 \pm 0.20}$	$\mathbf{93.91 \pm 0.23}$	$\mathbf{72.84 \pm 0.30}$	$\mathbf{89.87 \pm 0.19}$	$\mathbf{92.72 \pm 0.13}$	$\mathbf{92.65 \pm 0.11}$	$\mathbf{63.21 \pm 0.60}$

ness loss $\mathcal{L}_1$ promotes tight clusters, while degeneracy penalty $\mathcal{L}_2$ discourages small clusters:

$$\mathcal{L}_1 = \sum_{z=1}^{Z} \frac{1}{|\mathbb{C}_z|^2} \sum_{i,j \in \mathbb{C}_z} \mathcal{A}_{i,j}, \qquad \mathcal{L}_2 = \frac{1}{Z} \sum_{z=1}^{Z} \exp\left(\frac{\max\{0,\, \bar{\mathbb{C}} - \gamma\sigma_{\mathbb{C}} - |\mathbb{C}_z|\}}{\tau} \right) \tag{9}$$

where $\bar{\mathbb{C}} = N/Z$ and $\sigma_{\mathbb{C}}$ denote the mean and standard deviation of cluster sizes. A cluster $\mathbb{C}_z$ is penalized if size $|\mathbb{C}_z| < \bar{\mathbb{C}} - \gamma\sigma_{\mathbb{C}}$, with $\tau > 0$ controlling sharpness. The total loss is

$$\mathcal{L}_{\{\mathbb{C}_1,\ldots,\mathbb{C}_Z\}} = \mathcal{L}_1 + \lambda\mathcal{L}_2, \tag{10}$$

where $\lambda > 0$ balances the two terms. Lower $\mathcal{L}_1$ (tighter clusters) and $\mathcal{L}_2$ (less over-splitting) indicate better partitions. FEDDAG selects the clustering with the lowest loss and relatively few clusters.

4 GLOBAL REPRESENTATION SHARING (GRS)

High-level idea. In the previous section, we have combined data and gradient information to improve clustering. This section introduces global representation sharing across clusters during the training phase via a dual-encoder mechanism to further enhance FEDDAG's ability to learn complementary representations. The process for determining which clusters should complement each other and how training is carried out is described below:

Building Cluster Complementarity Graph (CC-Graph). We first determine which clusters can supply the class representation that others lack. Intuitively, a cluster has a *demand* for a class if that class is underrepresented among its clients, and a *supply* if the class is well represented. For class $c \in C$ we compute the *demand* of a requesting cluster $\mathbb{C}_p$ and the *supply* of a source cluster $\mathbb{C}_q$:

$$d_{p,c} = \sum_{i \in \mathbb{C}_p} (m_i - r_{i,c}), \quad s_{q,c} = \frac{1}{|\mathbb{C}_q|} \sum_{i \in \mathbb{C}_q} (r_{i,c} + 1), \tag{11}$$

where m_i is the number of distinct classes on client i and $r_{i,c} \in \{0, \ldots, m_i - 1\}$ is the rarity rank of class c on that client ($0 =$ rarest). Combining demand and supply yields the *complementarity score* between clusters p and q:

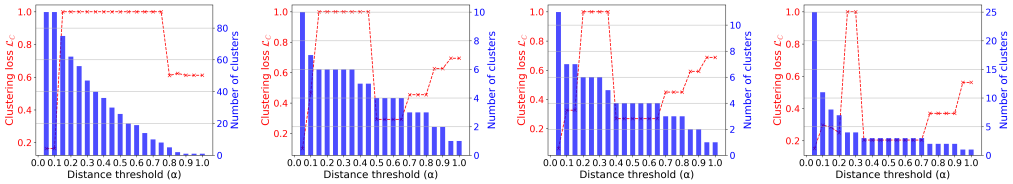
$$H_{p,q} = \sum_{c \in C} d_{p,c}\, s_{q,c}, \quad H_{p,p} = -\infty \tag{12}$$

Top-k values per row are retained to construct the adjacency matrix $H \in \{0,1\}^{Z \times Z}$, where an edge $p \to q$ indicates that cluster $\mathbb{C}_p$ receives representation from $\mathbb{C}_q$.

Training using dual encoders. For each client $i \in \mathbb{C}_z$, the prediction model can be described as:

$$F_z(X_i) = \psi\Big(\phi^{(1)}(X_i; \Theta_z^{1f}), \phi^{(2)}(X_i; \Theta_z^{2f})\,;\, \Theta_z^{c}\Big) \tag{13}$$

FEDDAG optimizes the parameters $\{\Theta_z^{1f}, \Theta_z^{2f}, \Theta_z^{c}\}_{z=1}^{Z}$ to minimize the weighted empirical loss across N clients. This is achieved through parallel training phases of the primary and secondary encoders. During the primary phase for each cluster, the primary encoder Θ_z^{1f} and the classifier Θ_z^{c} are optimized using data from clients $i \in \mathbb{C}_z$, enabling the model to learn its own cluster-specific features. In the secondary phase, clusters requesting knowledge from $\mathbb{C}_z$ first aggregate their secondary encoders Θ_j^{2f} and transmit the aggregated encoder to $\mathbb{C}_z$. The source cluster $\mathbb{C}_z$ then trains the received encoder on its local data and returns the resulting gradients to the requesting clusters for integration. The procedures for both phases and their unified training strategy are detailed below.



(a) CIFAR-10 (b) FMNIST (c) SVHN (d) CIFAR-100

Figure 1: Exp 2: Clustering score vs cluster α and number of clusters for finding optimal clustering.

(i) Primary encoder training. For each cluster, we optimize Θ_z^{1f} and Θ_z^c via gradient descent, while keeping the Θ_z^{2f} fixed. To approximate this, each client $i \in \mathbb{C}_z$ initializes $(\theta_i^{1f}, \theta_i^c) \leftarrow (\Theta_z^{1f}, \Theta_z^c)$ and keeps the secondary encoder Θ_z^{2f} frozen. The local loss is then defined as:

$$\ell_i(\theta_i^{1f}, \theta_i^c) = \mathcal{L}\big(Y_i, \psi(\phi^{(1)}(X_i; \theta_i^{1f}), \phi^{(2)}(X_i; \Theta_{z(i)}^{2f}); \theta_i^c)\big) \tag{14}$$

Using the client loss defined in Eq. 14, each client performs SGD training to update $(\theta_i^{1f}, \theta_i^c)$ as:

$$(\theta_i^{1f}, \theta_i^c) \leftarrow (\theta_i^{1f}, \theta_i^c) - \eta \nabla_{(\theta_i^{1f}, \theta_i^c)} \ell_i(\theta_i^{1f}, \theta_i^c), \ \forall i \in \mathbb{C}_z \tag{15}$$

FEDDAG aggregates the updates $(\theta_i^{1f} - \Theta_z^{1f})$ and $(\theta_i^c - \Theta_z^c)$ from client i to update $(\Theta_z^{1f}, \Theta_z^c)$ as:

$$\Theta_z^{1f} \leftarrow \Theta_z^{1f} + \sum_{i \in \mathbb{C}_z} \frac{|D_i|}{\sum_{k \in \mathbb{C}_z} |D_k|} (\theta_i^{1f} - \Theta_z^{1f}), \quad \Theta_z^c \leftarrow \Theta_z^c + \sum_{i \in \mathbb{C}_z} \frac{|D_i|}{\sum_{k \in \mathbb{C}_z} |D_k|} (\theta_i^c - \Theta_z^c) \tag{16}$$

(ii) Secondary encoder training. For each cluster $\mathbb{C}_z$, we optimize the secondary encoders $\{\Theta_j^{2f}\}$ of the clusters that seek to learn from $\mathbb{C}_z$. First, given the *CC-Graph* H, we first aggregate the secondary encoders of all learner clusters into a single combined encoder: $\Theta_z^{2f'} = \sum_{j\,:\,H(j,z)=1} \Theta_j^{2f}$. Then, each client $i \in \mathbb{C}_z$ initializes its local instance of the secondary encoder as $\theta_i^{2f'} \leftarrow \Theta_z^{2f'}$, while keeping Θ_z^{1f} and Θ_z^c fixed, and then minimizes the following loss:

$$\ell_i'(\theta_i^{2f'}) = \mathcal{L}\big(Y_i, \psi(\phi^{(1)}(X_i; \Theta_z^{1f}), \phi^{(2)}(X_i; \theta_i^{2f'}); \Theta_z^c)\big) \tag{17}$$

Using this loss, each client performs SGD to update as:

$$\theta_i^{2f'} \leftarrow \theta_i^{2f'} - \eta \nabla_{\theta_i^{2f'}} \ell_i'(\theta_i^{2f'}) \tag{18}$$

FEDDAG then aggregates the gradients $(\theta_i^{2f'} - \Theta_z^{2f'})$ from each client $i \in \mathbb{C}_z$, and update the secondary encoder of each learner cluster $\mathbb{C}_j$ (where $H(j, z) = 1$) as:

$$\Theta_j^{2f} \leftarrow \Theta_j^{2f} + \sum_{i \in \mathbb{C}_z} \frac{|D_i|}{\sum_{k \in \mathbb{C}_z} |D_k|} (\theta_i^{2f'} - \Theta_z^{2f'}) \tag{19}$$

Unifying Primary and Secondary Training. Since the primary and secondary encoder updates are independent (Eq. 15, 18), they can be trained in parallel. However, because the primary Θ_z^{1f} and secondary Θ_z^{2f} encoders are intended to capture complementary information, initializing them both randomly may lead to redundant features. To avoid this, we ensure the primary encoder is partially converged before joint training starts. Specifically, during gradient-based similarity computation in §3.1, each client i trains a local model to partial convergence. We reuse the resulting feature extractors θ_i^{0f} to initialize the global primary encoder Θ_z^{1f}, thereby avoiding extra training rounds:

$$\Theta_z^{1f} = \sum_{i \in \mathbb{C}_z} \frac{|D_i|}{\sum_{k \in \mathbb{C}_z} |D_k|} \theta_i^{0f}, \quad \forall z \in Z, \tag{20}$$

FEDDAG structure summary. During the initial rounds, FEDDAG determines the optimal clustering configuration (see Algorithm 1, Lines 1–14). Once the clustering is established, FEDDAG parallelly executes two phases: a primary training phase and a secondary global feature-sharing phase (Algorithm 1, Lines 15–23). Additional mechanisms for incorporating new clients and adapting to distribution shifts without interrupting training are provided in Appendix A.

5 EXPERIMENTS

This section experimentally evaluates FEDDAG, compares it against existing works, and investigates: (i) FEDDAG accuracy, (ii) Finding optimal clustering, (iii) Ablation studies, (iv) During evaluation, we report two variants of our method: FEDDAG*, which is restricted to the approach in §3—combining data and gradient information to form clusters and then training a standard clustered FL model (single encoder and classifier) without global representation sharing—and FEDDAG, which is the full algorithm that additionally incorporates dual-encoder inter-cluster sharing described in §4.

Table 3: Exp 3: Ablation study of cross-cluster representation sharing under 20% label skew (Dirichlet $\alpha' = 0.25$), comparing FEDDAG, FEDDAG† (dual encoder w/o GRS), and FEDDAG* (single encoder).

Algorithm	CIFAR-10	FMNIST	SVHN	CIFAR-100
FEDDAG†	88.79±0.20	92.61±0.31	91.95±0.25	70.28±0.38
FEDDAG*	88.67±0.18	92.75±0.22	91.87±0.26	70.37±0.33
FEDDAG	**90.76±0.12**	**93.82±0.20**	**93.91±0.23**	**72.84±0.30**

Baselines. We compare FEDDAG against SOTA methods: (i) single model FL: FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020), (ii) personalized FL method: PerFedAvg (Fallah et al., 2020), (iii) clustered FL — data-based: PACFL (Vahidian et al., 2023), (iv) clustered FL — gradient-based: IFCA (Ghosh et al., 2020), (CFL) (Sattler et al., 2020), FedSoft (Ruan & Joe-Wong, 2022), FedRC (Guo et al., 2024), FedGWC (Licciardi et al., 2025), CFL-GP (Kim et al., 2024a).

Table 4: Exp 5: Performance comparison for concept shift across datasets.

Algorithm	CIFAR-10	FMNIST	SVHN
FedAvg	42.87±0.36	42.68±0.49	37.93±0.39
FedSoft	64.34±0.38	75.89±0.15	76.35±0.40
PACFL	59.82±0.22	78.42±0.35	78.82±0.12
CFL	61.48±0.15	82.73±0.23	79.15±0.36
CFL-GP	66.74±0.28	84.71±0.13	82.38±0.13
FedGWC	65.91±0.19	83.85±0.21	81.63±0.28
FedRC	65.48±0.33	79.87±0.14	77.86±0.29
IFCA	64.58±0.39	84.67±0.21	81.56±0.14
FEDDAG*	67.79±0.27	86.03±0.21	83.73±0.19
FEDDAG	**69.13±0.23**	**88.79±0.19**	**85.06±0.26**

Experimental Setup. We consider 100 clients, with 20% randomly selected per round. Unless stated otherwise, all experiments run for 200 rounds with each selected client performing 10 local epochs (batch size 10, SGD). The principal vector U_c^i transmitted per class is roughly 1% the size of $|D_{i,c}|$. For gradient similarity $\mathcal{G}_{i,j}$, each client trains locally for t_g=2 rounds. To construct the *CC-Graph*, we select the top-k=2 source clusters.

Datasets. We use four popular datasets for the image classification task in FL setting, i.e., CIFAR-10 (Krizhevsky et al., 2009), FMNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), and CIFAR-100 (Krizhevsky et al., 2009).

Non-IID Data. We use multiple data distributions to simulate traditional and complex data skews:

• **Data Distribution I:** *This distribution evaluates* FEDDAG *under combined label skew and quantity shift.* To simulate label skew, we randomly select $\rho\%$ of labels and assign them to random client groups, repeating the process until all clients are assigned—similar to PACFL (Vahidian et al., 2023). For quantity shift, we allocate samples of the assigned labels using the Dirichlet factor (Ng et al., 2011). A real-world example is predictive text input, where users may discuss similar topics, but word distributions vary due to individual preferences and typing habits.

• **Data Distribution II:** *This distribution evaluates* FEDDAG *under concept shift.* Following prior work (Jothimurugesan et al., 2023; Guo et al., 2024), we simulate concept shift by modifying the labels of a subset of clients. For example, label y is changed to $(C-y)$ or $(y+1)\%C$, where C is the total number of classes. We perform three such transformations to simulate three distinct concepts. Similar modifications are applied to the test set.

• **Data Distribution III:** *This distribution evaluates* FEDDAG *under a different form of label skew.* We adopt the Latent Dirichlet Allocation (LDA) method from Hsu et al. (2019); Yurochkin et al. (2019), using Dirichlet concentration factors $\alpha' = 0.25$ and $\alpha' = 1.0$.

Additional experiments (e.g., performance evaluation, communication rounds) on the above and new distributions, hyperparameter selection and tuning, implementation details, ablation studies are provided in Appendix B. Algorithm theoretical issues, such as convergence, complexity, and privacy analysis; distribution and client shifts are discussed in Appendix A.

Experiments on Data Distribution I

Exp 1: Performance evaluation. We consider class skew $\rho = 20\%$ and 30%, with the Dirichlet concentration parameter α' set to *1* for low and *0.25* for high quantity shift. Table 2 shows the results for $\alpha' = 0.25$, while the results for $\alpha' = 1$ are included in Appendix B.3. We observe that

Table 5: Exp 5: Performance comparison under LDA skew ($\alpha' = 0.25$ and $\alpha' = 1.0$).

	$\alpha' = 0.25$			$\alpha' = 1.0$		
Algorithm	**CIFAR-10**	**FMNIST**	**SVHN**	**CIFAR-10**	**FMNIST**	**SVHN**
FedAvg	41.78 ± 0.73	47.26 ± 0.28	46.13 ± 0.48	66.48 ± 0.21	85.48 ± 0.36	81.89 ± 0.31
FedSoft	73.83 ± 0.42	83.75 ± 0.26	85.67 ± 0.19	71.08 ± 0.26	87.85 ± 0.31	85.92 ± 0.13
PACFL	80.52 ± 0.15	85.93 ± 0.12	87.23 ± 0.20	73.91 ± 0.43	93.31 ± 0.28	92.17 ± 0.23
CFL	78.94 ± 0.18	85.18 ± 0.17	85.19 ± 0.25	67.46 ± 0.12	83.16 ± 0.26	82.75 ± 0.28
CFL-GP	83.57 ± 0.15	86.43 ± 0.14	88.04 ± 0.19	73.84 ± 0.28	92.21 ± 0.23	91.67 ± 0.19
FedRC	81.76 ± 0.16	85.24 ± 0.22	87.91 ± 0.26	70.19 ± 0.42	88.27 ± 0.22	86.29 ± 0.42
IFCA	82.27 ± 0.19	87.53 ± 0.21	88.81 ± 0.13	74.43 ± 0.32	92.79 ± 0.33	92.12 ± 0.15
FEDDAG*	85.03 ± 0.21	89.65 ± 0.16	91.27 ± 0.22	75.52 ± 0.27	93.95 ± 0.20	93.08 ± 0.18
FEDDAG	$\mathbf{87.62 \pm 0.14}$	$\mathbf{91.88 \pm 0.10}$	$\mathbf{93.17 \pm 0.18}$	$\mathbf{77.84 \pm 0.23}$	$\mathbf{94.68 \pm 0.13}$	$\mathbf{94.15 \pm 0.11}$

single global FL baselines (e.g., FedAvg, FedProx) perform poorly under heterogeneity due to model drift (Zhao et al., 2018), while clustered FL methods yield stronger performance. Both variants of FEDDAG outperform state-of-the-art baselines—including data-based methods (e.g., PACFL) and gradient-based methods (e.g., IFCA, FedGWC). The lighter variant, FEDDAG*, achieves strong performance by combining data and gradient information to yield improved clustering. The full FEDDAG further enhances accuracy by enabling complementary representation sharing across clusters, allowing them to learn richer feature spaces.

Exp 2: Finding Optimal Cluster Formation. The server iterates over the clustering threshold α in Agglomerative HC at regular intervals (e.g., 0.05) to generate candidate clusterings. For each, the clustering loss $\mathcal{L}_{\{\mathbb{C}_1,\ldots,\mathbb{C}_Z\}}$ (see §3.4) is computed. In Figure 1, the x-axis shows α; the red curve indicates loss, and blue bars denote the number of clusters. Unlike traditional metrics (e.g., inertia) where loss decreases with more clusters, we observe abrupt increases in loss even as the number of clusters decreases for certain α values. This is due to FEDDAG's federated-aware clustering loss penalizing over-splitting into small clusters. The optimal α is selected as the point with low clustering loss and a relatively small number of clusters (e.g., for Figure 1(b) $\alpha^* = 0.65$).

Exp 3: Ablation Studies. We examine whether accuracy gains from inter-cluster global representation sharing (GRS) via the dual-encoder architecture (see §4) arise from genuine feature enrichment or simply from increased model parameters. To isolate this effect, we implement a dual-encoder variant with GRS disabled: during secondary-encoder training, instead of receiving representations from other clusters, each client trains its secondary encoder only on its own data and aggregates within its cluster. We denote this variant FEDDAG†; it is distinct from FEDDAG, which uses a single encoder. As shown in Table 3, full FEDDAG (with GRS) achieves the highest accuracy, while FEDDAG† performs comparably to FEDDAG, confirming that the gains of FEDDAG stem from cross-cluster representation sharing rather than model size alone.

Experiment on Data Distribution II

Exp 4: Performance under concept shift. Table 4 compares the performance of SOTA algorithms and FEDDAG on different datasets under concept shift and shows that FEDDAG achieves higher accuracy than the baselines. This improvement stems from FEDDAG's class-wise comparison mechanism, which provides more accurate similarity estimation under concept shift than existing methods.

Experiment on Data Distribution III

Exp 5: Performance under varying LDA skew. Table 5 shows accuracy under LDA-based skew with $\alpha' = 0.25$ and $\alpha' = 1.0$. FEDDAG consistently outperforms SOTA methods by leveraging cross-cluster feature sharing and integrating data and gradient information for clustering, leading to robust performance under LDA-based partition.

6 CONCLUSION

We develop a novel algorithm, FEDDAG, that addresses the limitations of existing clustered FL techniques and effectively tackles data heterogeneity challenges in FL by developing a novel method that combines both data and gradient information to cluster clients more effectively. Furthermore, FEDDAG utilizes representation sharing across clusters and incorporates an efficient mechanism to automatically determine the optimal number of clusters. Experiments on various heterogeneous data distributions demonstrate that FEDDAG outperforms existing approaches in terms of accuracy.

7 REPRODUCIBILITY STATEMENT

Code for the FEDDAG is included in the supplementary material. Additionally, convergence analysis of FEDDAG is provided in Appendix A.4.

REFERENCES

Luis B Almeida. Multilayer perceptrons. In *Handbook of Neural Computation*, pp. C1–2. CRC Press, 2020.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

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

Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 228–237. IEEE, 2021a.

Moming Duan, Duo Liu, Xinyuan Ji, Yu Wu, Liang Liang, Xianzhang Chen, Yujuan Tan, and Ao Ren. Flexible clustered federated learning for client-level data distribution shift. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2661–2674, 2021b.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in neural information processing systems*, 33:3557–3568, 2020.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5736–5745, 2017.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

Yongxin Guo, Xiaoying Tang, and Tao Lin. Fedrc: tackling diverse distribution shifts challenge in federated learning by robust clustering. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 665–674, 2013.

Meirui Jiang, Anjie Le, Xiaoxiao Li, and Qi Dou. Heterogeneous personalized federated learning by local-global updates mixing via convergence rate. In *The Twelfth International Conference on Learning Representations*, 2024.

Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B Gibbons. Federated learning under distributed concept drift. In *International Conference on Artificial Intelligence and Statistics*, pp. 5834–5853. PMLR, 2023.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Myeongkyun Kang, Soopil Kim, Kyong Hwan Jin, Ehsan Adeli, Kilian M Pohl, and Sang Hyun Park. Fednn: Federated learning on concept drift data using weight and adaptive group normalizations. *Pattern Recognition*, 149:110230, 2024.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

Heasung Kim, Hyeji Kim, and Gustavo De Veciana. Clustered federated learning via gradient-based partitioning. In *Forty-first International Conference on Machine Learning*, 2024a.

Heasung Kim, Hyeji Kim, and Gustavo De Veciana. Clustered federated learning via gradient-based partitioning. In *Forty-first International Conference on Machine Learning*, 2024b.

Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

Minghao Li, Dmitrii Avdiukhin, Rana Shahout, Nikita Ivkin, Vladimir Braverman, and Minlan Yu. Federated learning clients clustering with adaptation to data drifts. *arXiv preprint arXiv:2411.01580*, 2024.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Alessandro Licciardi, Davide Leo, Eros Faní, Barbara Caputo, and Marco Ciccone. Interaction-aware gaussian weighting for clustered federated learning. *arXiv preprint arXiv:2502.03340*, 2025.

Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. Multi-center federated learning: clients clustering for better personalization. *World Wide Web*, 26(1):481–500, 2023.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34:15434–15447, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Kai Wang Ng, Guo-Liang Tian, and Man-Lai Tang. Dirichlet and related distributions: Theory, methods and applications. 2011.

Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022a.

Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pp. 17716–17758. PMLR, 2022b.

Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 8124–8131, 2022.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

Saeed Vahidian, Mahdi Morafah, Weijia Wang, Vyacheslav Kungurtsev, Chen Chen, Mubarak Shah, and Bill Lin. Efficient distribution similarity identification in clustered federated learning via principal angles between client data subspaces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 10043–10052, 2023.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Addressing class imbalance in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10165–10173, 2021.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Yue Wu, Shuaicheng Zhang, Wenchao Yu, Yanchi Liu, Quanquan Gu, Dawei Zhou, Haifeng Chen, and Wei Cheng. Personalized federated learning under mixture of distributions. In *International Conference on Machine Learning*, pp. 37860–37879. PMLR, 2023.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*, pp. 7252–7261. PMLR, 2019.

Dun Zeng, Xiangjing Hu, Shiyu Liu, Yue Yu, Qifan Wang, and Zenglin Xu. Stochastic clustered federated learning. *arXiv preprint arXiv:2303.00897*, 2023.

Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. Federated learning with label distribution skew via logits calibration. In *International Conference on Machine Learning*, pp. 26311–26329. PMLR, 2022.

Yuxin Zhang, Haoyu Chen, Zheng Lin, Zhe Chen, and Jin Zhao. Fedac: A adaptive clustered federated learning framework for heterogeneous data. *arXiv preprint arXiv:2403.16460*, 2024.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

A TECHNICAL DISCUSSION AND ANALYSIS

This section presents a detailed discussion and breakdown of FEDDAG, covering key design elements, communication and privacy considerations, adaptability to new clients and shifting distributions, and practical implementation details.

A.1 ADDITIONAL RELATED WORK

FL with Heterogeneous Data. Handling data heterogeneity remains a fundamental challenge in federated learning, as clients often hold non-IID datasets that degrade the performance of standard aggregation schemes. Personalized FL methods Fallah et al. (2020); Liang et al. (2020); Smith et al. (2017); Arivazhagan et al. (2019) aim to tailor models to individual clients improving local accuracy while still benefiting from partial knowledge sharing. Aggregation-based approaches Wang et al. (2020); Pillutla et al. (2022a); Karimireddy et al. (2020) modify the server-side model update to mitigate client drift caused by non-IID data, often using correction terms or robust optimization techniques. Local–global mixing strategies Jiang et al. (2024); Mansour et al. (2020); Deng et al. (2020) combine local model training with global knowledge transfer, balancing personalization and collaboration to better handle skewed distributions.

A.2 PRELIMINARIES

Principal Angles Between Two Subspaces. Consider two subspaces, $\mathcal{V} = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_p\}$ and $\mathcal{X} = \text{span}\{\mathbf{x}_1, \ldots, \mathbf{x}_q\}$, where $\mathcal{V}$ and $\mathcal{X}$ are p-dimensional and q-dimensional subspaces of $\mathbb{R}^n$, respectively. The sets $\{\mathbf{v}_1, \ldots, \mathbf{v}_p\}$ and $\{\mathbf{x}_1, \ldots, \mathbf{x}_q\}$ are orthonormal, with $1 \leq p \leq q$. A sequence of p principal angles, $0 \leq \Phi_1 \leq \Phi_2 \leq \cdots \leq \Phi_p \leq \frac{\pi}{2}$, is defined to measure the similarity between the subspaces. These angles are calculated as:

$$\Phi(\mathcal{V}, \mathcal{X}) = \min_{\mathbf{v} \in \mathcal{V}, \mathbf{x} \in \mathcal{X}} \cos^{-1} \left(\frac{|\mathbf{v}^T \mathbf{x}|}{\|\mathbf{v}\| \|\mathbf{x}\|} \right) \tag{21}$$

where $\| \cdot \|$ is the norm. The smallest of these angles is $\Phi_1(\mathbf{v}_1, \mathbf{x}_1)$, with the vectors $\mathbf{v}_1$ and $\mathbf{x}_1$ as the corresponding principal vectors. The principal angle distance serves as a metric to quantify the separation between subspaces Jain et al. (2013).

Agglomerative hierarchical clustering (HC). (Day & Edelsbrunner, 1984) is a popular method in machine learning for grouping similar objects based on an adjacency (proximity) matrix. We found HC to be the best fit for FEDDAG. We also experimented with other clustering algorithms, e.g., K-means and graph clustering, but we observed that the clustering algorithm does not make much difference in cluster formation. HC begins by treating each data point as its own cluster. During each iteration, HC identifies two clusters that are most similar and merges them. The criterion for selecting which clusters to merge depends on a linkage method; e.g., in *single linkage*, the L_2 (Euclidean) distance between two clusters is defined as the smallest distance between any pair of points from the two clusters. As a merging criterion, FEDDAG defines a ***clustering threshold*** $\boldsymbol{\alpha \in (0, 1]}$, such that any two clusters with a distance less than α are merged.; e.g., α=1 results in all clients being grouped into a single cluster.

A.3 FEDDAG OVERVIEW & ALGORITHMS

An illustration of the FEDDAG algorithm is shown in Figure 2. The algorithm for class-wise weighted data-based similarity computation is shown in Algorithm 2. And, the algorithm for combining both data and gradient information to improve clustering is shown in Algorithm 3.

A.4 CONVERGENCE ANALYSIS

Following Pillutla et al. (2022b) that works on partial model personalization, we consider the shared–personalized objective:

$$\min_{u, V} \; F(u, V) \; := \; \frac{1}{n} \sum_{i=1}^{n} F_i\big(u, v_i\big), \tag{22}$$

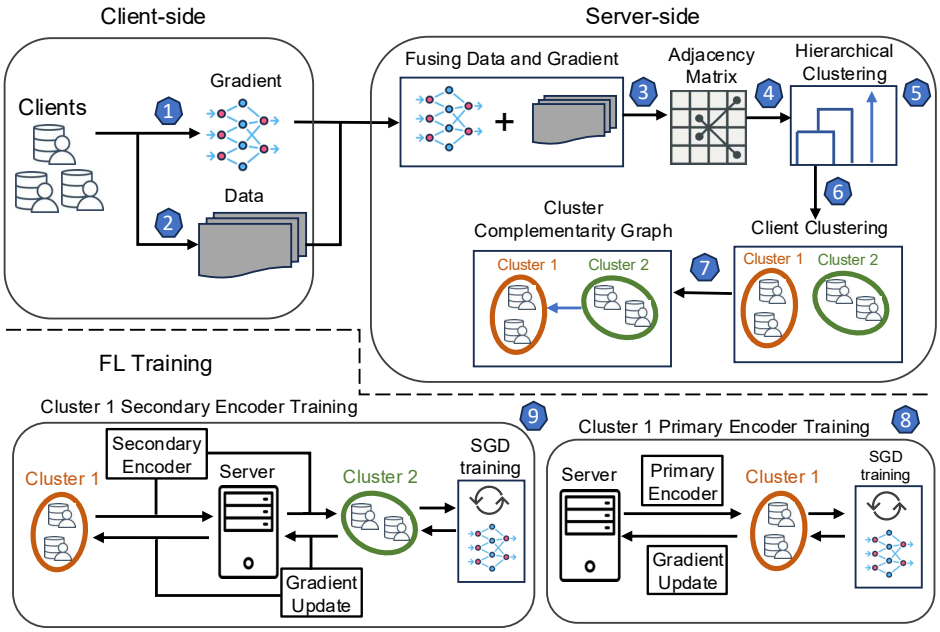


Figure 2: Overview of FEDDAG. Clients compute principal vectors and gradients, which the server uses to build an adjacency matrix via hierarchical clustering. A cluster complementarity graph then indicates which clusters can supply features for cross-cluster sharing. Training proceeds in two phases: (1) the primary encoder and classifier are trained on each cluster's local data; (2) the secondary encoder of a requesting cluster is sent to a source cluster, trained with its data, and returned as gradients for integration.

where u denotes shared parameters and $V = \{v_i\}_{i=1}^{n}$ personalized parameters. In our dual-encoder model (Eq. equation 13), for each cluster z we map the *secondary encoder* as the *shared* block and the *primary encoder* (optionally together with the classifier) as the *personalized* block:

$$u_z \longmapsto \Theta_z^{2f} \quad \text{(shared: secondary encoder)},$$

$$V_z \longmapsto (\Theta_z^{1f}, \Theta_z^{c}) \quad \text{(personalized: primary encoder + classifier)}.$$

Given a fixed clustering $\{\mathbb{C}_z\}_{z=1}^{Z}$ (one-shot data and gradient combined similarity; see §3), the cluster-level empirical risk can be written in the shared–personalized form of Pillutla et al. (2022b):

$$\min_{\{u_z, V_z\}_{z=1}^{Z}} F(\{u_z, V_z\}) = \sum_{z=1}^{Z} \sum_{i \in \mathbb{C}_z} \frac{|D_i|}{\sum_{k \in \mathbb{C}_z} |D_k|} F_i(u_z, V_z),$$

$$F_i(u_z, V_z) = \mathcal{L}\Big(Y_i,\, \psi\big(\phi^{(1)}(X_i; \Theta_z^{1f}),\, \phi^{(2)}(X_i; u_z); \Theta_z^{c}\big)\Big).$$

Thus, for each cluster z, Pillutla et al. (2022b)'s analysis applies to the pair (u_z, V_z), and the full objective is a weighted sum over clusters. So, based on this, we will define notations, assumptions, and the convergence analysis below:

Block notation and participation model. For each cluster $z \in \{1, \ldots, Z\}$ in the fixed partition $\{\mathbb{C}_z\}_{z=1}^{Z}$, we decompose the parameters as

$$u_z := \Theta_z^{2f} \qquad \text{(cluster–global / secondary encoder)}, \tag{23}$$

$$V_z := (\Theta_z^{1f}, \Theta_z^{c}) \qquad \text{(cluster–personal: primary encoder + classifier)}. \tag{24}$$

Let m be the total number of clients and $m_z := |\mathbb{C}_z|$ the number of clients in cluster z; define the cluster weights

$$\pi_z := \frac{m_z}{m}, \qquad \sum_{z=1}^{Z} \pi_z = 1. \tag{25}$$

In each communication round, cluster z samples s_z clients (without replacement) and runs E local steps. The average per-round participation fraction is

$$\bar{q} \;:=\; \sum_{z=1}^{Z} \pi_z \, \frac{s_z}{m_z} \in (0,1]. \tag{26}$$

Loss and per-client objective. For client $i \in \mathbb{C}_z$ with data $D_i = (X_i, Y_i)$, define

$$F_i(u_z, V_z) \;:=\; \mathcal{L}\Big(Y_i,\, \psi\big(\phi^{(1)}(X_i; \Theta_z^{1f}),\, \phi^{(2)}(X_i; u_z); \Theta_z^{c}\big)\Big), \tag{27}$$

and the cluster-weighted empirical risk

$$F\big(\{u_z, V_z\}_{z=1}^{Z}\big) \;:=\; \sum_{z=1}^{Z} \sum_{i \in \mathbb{C}_z} \frac{|D_i|}{\sum_{k \in \mathbb{C}_z} |D_k|}\; F_i(u_z, V_z). \tag{28}$$

Assumptions used in the theorem. We state the standard conditions in our block notation; expectations are w.r.t. the algorithm's sampling and stochasticity.

Assumption A.1 (Smoothness)**.** *Each client loss in equation 27 is L-smooth in (u_z, V_z). For all (u_z, V_z) and (u'_z, V'_z),*

$$\big\|\nabla_{(u_z,V_z)} F_i(u_z, V_z) - \nabla_{(u_z,V_z)} F_i(u'_z, V'_z)\big\| \;\le\; L\,\big\|(u_z, V_z) - (u'_z, V'_z)\big\|. \tag{29}$$

Equivalently, F_i is L-smooth in each sub-block Θ_z^{1f}, Θ_z^{2f}, and Θ_z^{c}.

Assumption A.2 (Unbiased stochastic gradients with bounded variance)**.** *For any sampled client $i \in \mathbb{C}_z$,*

$$\mathbb{E}\big[\tilde{\nabla}_{u_z} F\big] = \nabla_{u_z} F, \qquad \mathbb{E}\big[\|\tilde{\nabla}_{u_z} F - \nabla_{u_z} F\|^2\big] \le \sigma_{u,z}^2, \tag{30}$$

$$\mathbb{E}\big[\tilde{\nabla}_{V_z} F\big] = \nabla_{V_z} F, \qquad \mathbb{E}\big[\|\tilde{\nabla}_{V_z} F - \nabla_{V_z} F\|^2\big] \le \sigma_{V,z}^2, \tag{31}$$

where $\nabla_{V_z} F := (\nabla_{\Theta_z^{1f}} F, \nabla_{\Theta_z^{c}} F)$. Define the cluster-weighted variances

$$\bar{\sigma}_u^2 := \sum_{z=1}^{Z} \pi_z\, \sigma_{u,z}^2, \qquad \bar{\sigma}_V^2 := \sum_{z=1}^{Z} \pi_z\, \sigma_{V,z}^2. \tag{32}$$

Assumption A.3 (Gradient diversity / heterogeneity)**.** *Let $F_z(u_z, V_z) := \frac{1}{\sum_{k \in \mathbb{C}_z} |D_k|} \sum_{i \in \mathbb{C}_z} |D_i|\, F_i(u_z, V_z)$ be the average loss in cluster z. There exist finite constants $\delta_{\text{in}}^2 \ge 0$ and $\delta_{\text{out}}^2 \ge 0$ such that*

$$\sum_{z=1}^{Z} \pi_z \left\|\nabla_{u_z} F_z - \nabla_{u_z} F\right\|^2 \le \delta_{\text{in}}^2, \qquad \sum_{z=1}^{Z} \pi_z \left\|\nabla_{V_z} F_z - \nabla_{V_z} F\right\|^2 \le \delta_{\text{in}}^2, \tag{33}$$

and the cross-cluster mismatch (relevant to the sharing step) is bounded by δ_{out}^2.

Assumption A.4 (Stable clustering)**.** *The partition $\{\mathbb{C}_z\}_{z=1}^{Z}$ obtained at initialization (t=0) is fixed for the entire analysis horizon $t = 1, \ldots, T$: no clients are reassigned, and clusters do not split or merge.*

Assumption A.5 (Cross-cluster sharing noise)**.** *The cross-cluster representation sharing (via the CC-Graph) is either deterministic (no additional noise), or it introduces an additive variance bounded by σ_{share}^2 in the updates of the u-blocks.*

Initial suboptimality. We denote the initial gap by

$$\Delta_\ell \;:=\; F\big(\{u_z^0, V_z^0\}_{z=1}^{Z}\big) - F^\star, \tag{34}$$

where $F^\star$ is the optimal value of equation 28.

Theorem A.1 (Convergence of FEDDAG (per-cluster globals, dual encoders))**.** *Let the assumptions above hold. Choose learning rates $\eta = \tau/(LE)$ and $\eta_{\text{share}} = \Theta(1/L)$, for a constant τ depending on L, the variance terms, heterogeneity, and participation. Then, ignoring absolute constants and provided clustering is stable,*

$$\frac{1}{T}\sum_{t=1}^{T}\left[\frac{1}{L}\sum_{z=1}^{Z}\mathbb{E}\big\|\nabla_{u_z} F\big\|^2 \;+\; \frac{1}{mL}\sum_{i=1}^{m}\mathbb{E}\big\|\nabla_{V_{c(i)}} F\big\|^2\right] \;\le\; \frac{(\Delta_\ell\, \sigma_{\text{sim},1}^2)^{1/2}}{T^{1/2}} + \frac{(\Delta_\ell^2\, \sigma_{\text{sim},2}^2)^{1/3}}{T^{2/3}} + \mathcal{O}\Big(\frac{1}{T}\Big), \tag{35}$$

where the effective variance terms are

$$\sigma^2_{\text{sim},1} \;=\; \frac{2}{L}\left(\delta^2_{\text{in}}\sum_{z=1}^{Z}\pi_z\Big(1-\frac{s_z}{m_z}\Big) \;+\; \frac{\bar{\sigma}^2_u}{L} \;+\; \sum_{z=1}^{Z}\pi_z\,\frac{s_z}{m_z}\,\sigma^2_{V,z} \;+\; \sigma^2_{\text{share}}\right), \tag{36}$$

$$\sigma^2_{\text{sim},2} \;=\; \frac{2}{L}\left(\delta^2_{\text{in}}+\delta^2_{\text{out}} \;+\; \bar{\sigma}^2_u+\bar{\sigma}^2_V \;+\; \sigma^2_{\text{share}}\right)\Big(1-\frac{1}{E}\Big). \tag{37}$$

Remark 1 (Clustering stability). The bound relies on a fixed partition; oscillations due to re-clustering invalidate the descent decomposition. In practice, stability is supported empirically by (i) one-shot blended (data+gradient) clustering at $t{=}0$ and (ii) the fact that training is conducted within the fixed clusters thereafter.

Remark 2 (What differs vs. single-global frameworks). In FEDDAG, we aggregate *both* the per-cluster global blocks $u_{1:Z}$ (secondary encoders, coupled via the CC-Graph) and the cluster-personal blocks $V_{1:Z}$ (primary encoder + classifier). Consequently, $\sigma^2_{\text{sim},1}$ and $\sigma^2_{\text{sim},2}$ expose: (i) per-cluster sampling s_z/m_z (larger s_z improves the first term), (ii) local steps E (fewer local steps reduce the drift factor $1-1/E$), and (iii) cross-cluster sharing noise σ^2_{share} (zero for deterministic Laplacian smoothing; small but positive for stochastic distillation). The asymptotic $T^{-1/2}$ rate is observed once all devices are seen on average at least once; a convenient sufficient condition (up to constants) is

$$T \;\geq\; \frac{\Delta_\ell}{\sigma^2_{\text{sim},1}}\;\max\left\{\frac{(1-\bar{q})\,E}{\bar{q}},\;2\right\}, \qquad \bar{q}=\sum_{z=1}^{Z}\pi_z\,\frac{s_z}{m_z}. \tag{38}$$

A.5 Communication and Computation Complexity

FEDDAG minimizes communication and computation overhead, aligning with the scalability requirements of federated learning systems. Before dual-encoder joint training begins, each client locally trains for t_g rounds without federation (see Algorithm 1). At the end of this phase, each client uploads: (i) a k-sparse gradient $\tilde{\Delta}_i$ of dimension $k \ll |D_i|$, and (ii) class-wise p principal vectors $U^i_c \in \mathbb{R}^{d\times r}$ for $c=1,\ldots,C$. The number of principal vectors p is kept small (typically 1–2% of the class size). Hence, the combined communication cost of $\tilde{\Delta}_i$ and U^i_c is negligible relative to the size of the model parameter space $|\Theta|$. The computation of principal vectors via SVD incurs a cost of $\mathcal{O}(FN^2)$ per client, assuming a local dataset of N samples and F features with $N>F$.

Once the proximity matrix and clustering are finalized, FEDDAG maintains the same per-round communication cost as FedAvg in terms of transmitting model parameters. However, due to its dual-encoder architecture, it additionally transmits a secondary encoder (Θ^{2f}) alongside the primary encoder (Θ^{1f}), both of equal size. In each training round, selected clients perform two local SGD phases:

- *Primary phase* — standard local update on (θ^{1f},θ^c).
- *Secondary phase* — additional local update on $\theta^{2f'}$, which has the same size as θ^{1f}.

If both phases are executed in the same round, the local computation cost is approximately $2\times$ that of FedAvg. However, the two phases can be alternated, with each running every other round in settings where computation is constrained. Since updates to the primary and secondary encoders are independent, the correctness and convergence of the final model are preserved under this alternating schedule.

A.6 Privacy Considerations

Privacy is a foundational aspect of federated learning, which aims to enable collaborative model training while protecting the sensitive data of individual clients. In the context of FEDDAG, we examine the privacy implications of both the similarity estimation and representation-sharing phases. During client clustering, FEDDAG constructs a weighted, class-wise data similarity matrix using a small set of class-representative principal vectors and per-class sample counts provided by each client. Crucially, the shared principal vectors are reduced linear combinations of local data and do

Algorithm 2: Proximity Matrix Computation

Input: Principal vectors U^*, sparsified gradients $\tilde{\Delta}^*$ for all clients
Output: $\mathcal{A}$, proximity matrix between all client pairs
Function: `ProximityMatrix`$(U^*, \tilde{\Delta}^*)$
for *client* $i = 1, \ldots, N$ and **for** *client* $j = 1, \ldots, N$ **do**
  for *class* $c = 1, \ldots, C$ **do**
    Compute $\mathcal{V'}_{i,j,c}$ using Eq. 4
    Compute $\mathcal{W'}_{i,j,c}$ using Eq. 5
  Compute $\mathcal{V}_{i,j}$ using Eq. 6 and $\hat{\mathcal{V}} \leftarrow$ normalize$(\mathcal{V})$
  Compute $\mathcal{G}_{i,j}$ using Eq. 3 and $\hat{\mathcal{G}} \leftarrow$ normalize$(\mathcal{G})$
Initialize weight vector $\mathbf{w} = (w_1, \ldots, w_N)^\top \in [0,1]^N$ randomly
while *not converged* **do**
  Compute entropy loss $\mathcal{L}_{\text{en}}$ using Eq. 8
  $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}_{\text{en}}$; and $\mathbf{w} \leftarrow$ clip$(\mathbf{w}, 0, 1)$ `// MLP-based update`
Compute $A_{i,j}$ as in Eq. 7 and **return** $\mathcal{A}_{i,j}$

Algorithm 3: Clustering Threshold Search in FL

Input: Proximity matrix $\mathcal{A}_{i,j}$, threshold set S_α
Output: Optimal clustering $\{\mathbb{C}_1, \ldots, \mathbb{C}_Z\}$
Function `OptimalClustering`$(\mathcal{A}, S_\alpha)$**:**
  Initialize empty list `records`
  for $\alpha \in S_\alpha$ **do**
    Generate candidate clustering $\mathbb{C}^\alpha$ using hierarchical clustering (HC) on $\mathcal{A}$ with threshold α
    Compute $\mathcal{L}_1$ and $\mathcal{L}_2$ (Eq. 9) for $\mathbb{C}^\alpha$
    Total clustering score $\mathcal{L}_{\{\mathbb{C}_1,\ldots,\mathbb{C}_Z\}} = \mathcal{L}_1 + \lambda \mathcal{L}_2$
    Save tuple $(\alpha, \mathcal{L}_{\{\mathbb{C}_1,\ldots,\mathbb{C}_Z\}})$ to `records`
  Select α^* with low score and relatively small Z from `records`
  return Optimal clustering $\{\mathbb{C}_1, \ldots, \mathbb{C}_Z\} \leftarrow \mathbb{C}^{\alpha^*}$

not expose any raw samples or labels. Moreover, each client contributes fewer than 1% of such vectors per class, ensuring minimal data exposure. This approach aligns with prior privacy-aware clustering methods Vahidian et al. (2023), which also transmit low-dimensional representative vectors to the server. In more privacy-sensitive deployments, additional protection mechanisms can be integrated into FEDDAG. For instance, secure aggregation protocols (Bonawitz et al., 2017), encryption techniques, or differential privacy can be used to protect the shared principal vectors. Additionally, uniform weighting can be employed in place of class-frequency-based weighting of similarity values to prevent leakage of class distribution information. To further mitigate information leakage during gradient-based similarity estimation, FEDDAG can adopt encryption strategies similar to those proposed in Sattler et al. (2020).

During cross-cluster feature sharing (see §4), when a cluster requests representations from a source cluster, only the aggregated gradients computed from the source cluster's clients are shared. No individual client's gradient information is exposed at any point.

A.7 Generalization to Newcomers — Algorithms 4

High-level idea. In real-world FL systems, new clients may join after the initial clustering and model training have already begun. Moreover, clients may not always remain continuously available. To handle such cases, we extend FEDDAG with a lightweight mechanism that allows new clients to seamlessly join existing clusters without disrupting ongoing training. Specifically, each new client computes its data and gradient information, which are used to extend the proximity matrix to include similarity values for the new client. This updated matrix is then used by the clustering algorithm to determine the appropriate cluster assignment. Once assigned, the client is integrated into the designated cluster without re-evaluating the optimal clustering or retraining any previously learned weights.

Details of the method. The process for integrating a new client i_{new} is similar to that used for initial clients (as in §3). FEDDAG first performs local training on i_{new}'s data for t_{g} rounds to reach partial convergence. Afterwards, client i_{new} computes its sparsified gradient update $\tilde{\Delta}^{i_{\text{new}}}$ and class-wise principal vectors $U_c^{i_{\text{new}}}$ and sends them to the server. The server updates the existing data similarity matrix $\hat{\mathcal{V}}_{i,j}$ and gradient similarity matrix $\hat{\mathcal{G}}_{i,j}$ to their extended forms $\hat{\mathcal{V}}_{i,j}^{\text{new}}$ and $\hat{\mathcal{G}}_{i,j}^{\text{new}}$, incorporating information from the new client. To combine the data and gradient, FEDDAG initially learns a weight vector $\mathbf{w}$ (see §3.3). To integrate the new client, FEDDAG extends this process by assigning

Algorithm 4: Generalization to Newcomers

Input: New client i_{new}, clustering threshold α^*, current clusters $\{\mathbb{C}_1, \ldots, \mathbb{C}_Z\}$, current proximity matrix $\mathcal{A}_{i,j}$, data matrix $\mathcal{V}_{i,j}$, gradient matrix $\mathcal{G}_{i,j}$

Output: Updated client i_{new} models

1 **Function** `NewcomerIntegration(`i_{new}, α^*`)`**:**
2 Initialize client i_{new} with random $\theta^0_{i_{\text{new}}}$
3 Set local counter $t_{i_{\text{new}}} = 0$
`// Tracks local warm-up rounds`
4 **for** *each global round* t **do**
5 **if** $t_{i_{new}} < t_g$ **then**
6 Local training of $\theta^0_{i_{\text{new}}}$ using local data (no federation)
7 $t_{i_{\text{new}}} \leftarrow t_{i_{\text{new}}} + 1$
8 **if** $t_{i_{new}} = t_g$ **then**
9 Client i_{new} sends $\tilde{\Delta}^{i_{\text{new}}}$ and $U_c^{i_{\text{new}}}$ to server
`// --- Extend proximity matrix ---`
10 Server extends $\hat{\mathcal{V}}^{\text{new}}_{i_{\text{new}},j}$ and $\hat{\mathcal{G}}^{\text{new}}_{i_{\text{new}},j}$ to include the new client
11 Server initializes $w_{i_{\text{new}}} \in [0, 1]$ and learns it using Eq. 8 (§3), keeping existing weights fixed
12 Server extends proximity matrix $\mathcal{A}^{\text{new}}_{i,j}$ using Eq. 7 (§3) with $\hat{\mathcal{V}}^{\text{new}}_{i,j}$ and $\hat{\mathcal{G}}^{\text{new}}_{i,j}$
`// --- Cluster assignment ---`
13 Server executes hierarchical clustering with α^* on $\mathcal{A}^{\text{new}}_{i,j}$ to assign i_{new} to cluster $\mathbb{C}_{z(i_{\text{new}})}$
14 Client i_{new} sets $\theta^{1f}_{i_{\text{new}}}, \theta^c_{i_{\text{new}}}$ from $(\Theta^{1f}_{z(i_{\text{new}})}, \Theta^c_{z(i_{\text{new}})})$
`// Aggregate secondary encoders from related clusters (via` $\mathcal{H}$`)`
15 Client i_{new} sets $\theta^{2f'}_{i_{\text{new}}} \leftarrow \sum_{j:H(j,z(i_{\text{new}}))=1} \Theta^{2f}_j$
16 **else**
`// --- Standard training phase (same as` **`else`** `branch in Algorithm 1) ---`
17 Train $(\theta^{1f}_{i_{\text{new}}}, \theta^c_{i_{\text{new}}})$ via SGD using Eq. 15
`// Primary training`
18 Train $\theta^{2f'}_{i_{\text{new}}}$ via SGD using Eq. 18
`// Secondary training`
19 Broadcast updated $(\theta^{1f}_{i_{\text{new}}}, \theta^c_{i_{\text{new}}}, \theta^{2f'}_{i_{\text{new}}})$ to server

a weight $w_{i_{\text{new}}} \in [0, 1]$ and learning it using the same entropy loss (Eq. 8) used during initial training, but optimizing only for $w_{i_{\text{new}}}$ without modifying existing weights. Using the extended weights $\mathbf{w}_{\text{new}}$, the proximity matrix $\mathcal{A}^{\text{new}}_{i,j}$ is computed based on $\hat{\mathcal{V}}^{\text{new}}_{i,j}$, $\hat{\mathcal{G}}^{\text{new}}_{i,j}$, and $w_{i_{\text{new}}}$, as defined in Eq. 7 in §3.3.

Finally, the server reuses the previously selected clustering threshold α^* (from Optimal Clustering in §3.4) and performs a single hierarchical clustering (HC) pass on $\mathcal{A}^{\text{new}}_{i,j}$ to assign i_{new} to a cluster $\mathbb{C}_{z(i_{\text{new}})}$. After assignment, client i_{new} initializes its model from the corresponding cluster's global parameters and directly joins the existing FEDDAG training flow (i.e., the **else** branch at line 15 in Algorithm 1). This extension enables efficient onboarding of new clients by reusing the established clustering threshold and global models, avoiding disruption to ongoing training. The complete process is summarized in Algorithm 4. Also, we evaluate the generalization capability of FEDDAG to unseen clients through experiments reported in Appendix §B.6.

A.8 HANDLING DATA-DISTRIBUTION SHIFT

High-level idea. After FEDDAG has converged, the data of already-clustered clients may still evolve over time (e.g., new sensor drifts, changes in user behavior). If the local distribution of a client drifts too far from what its current cluster represents, the global model quality may degrade. We, therefore, add a mechanism that decides whether a client needs to be re-evaluated for cluster assignment. In addition, to accommodate a growing client population, FEDDAG periodically re-assesses the clustering to ensure the configuration remains consistent with the evolving client landscape. Specifically, the Wasserstein distance (Duan et al., 2021b) is employed to track shifts in the class distribution of each client's local data over time; when a significant shift is detected, the system recomputes that client's data and gradient representations and re-evaluates its proximity to other clients using the same similarity fusion mechanism described in §3. This enables re-clustering of the client without disrupting other participants or restarting global training.

Client re-evaluation. Let $\mathcal{P}_i^{(t)}$ denote the empirical class histogram of client i at round t. Every δ' rounds, we compute the 1-Wasserstein distance[3] between the current and previous histograms as

[3]For image classification, we treat classes as discrete points on the line $0, \ldots, C-1$; the 1-Wasserstein distance then has a closed form based on cumulative histograms.

$W_1\big(\mathcal{P}_i^{(t)}, \mathcal{P}_i^{(t-\delta')}\big)$. Client i is marked as shifted if:

$$W_1\big(\mathcal{P}_i^{(t)}, \mathcal{P}_i^{(t-\delta')}\big) \;>\; \tau_i \;:=\; \frac{0.2}{\textit{LabelSize}} \cdot n_i, \tag{39}$$

where n_i is the number of new samples processed by client i since round $t-\delta'$. Eq. 39 flags a shift when roughly 20% of local data has changed.

A shifted client does not immediately trigger global re-clustering. Instead, its cluster assignment is re-evaluated through a procedure that re-computes data and gradient information for the server (similar to generalizing to newcomer clients in §A.7). For the next t_g rounds, the shifted client i trains its primary encoder and classifier on local data *without federation* so that the resulting gradients reflect its own distribution rather than the global model. After local training, the client computes its gradient update Δ^i, applies k-sparsification to obtain $\tilde{\Delta}^i$, re-computes class-wise principal vectors U_c^i from local data, and sends U_c^i and $\tilde{\Delta}^i$ to the server. These components update the data similarity matrix $\hat{\mathcal{V}}_{i,j}$, the gradient similarity matrix $\hat{\mathcal{G}}_{i,j}$, and the proximity matrix $\mathcal{A}_{i,j}$. The server then re-evaluates clients' cluster assignments by performing a single hierarchical-clustering pass on the updated $\mathcal{A}_{i,j}$ using the fixed optimal threshold α^* (derived in §3.4). If a reassignment occurs, the client initializes its model from the corresponding global model and continues training.

Accommodating growing population. To support an expanding set of participants, FEDDAG periodically re-evaluates the clustering after a specified number of new clients have joined. This reassessment determines whether the updated client distribution warrants a change in the cluster structure. Concretely, FEDDAG re-runs the optimal clustering selection procedure by sweeping over candidate threshold values α (as in §3.4). If a new clustering configuration is chosen, the algorithm updates the necessary components (e.g., the cluster complementarity graph, re-initializes the global model from client models) and resumes training, ensuring consistency with the evolving client landscape.

B Additional Experiments

In this section, we show implementation details, additional experiments regarding hyperparameter selection and sensitivity, and FEDDAG performance on different data distributions.

B.1 Implementation Details

We now describe the implementation details used in our experiments, including model architectures and training hyperparameters. For datasets such as CIFAR-10 and SVHN, we adopt a convolutional neural network (LeCun et al., 2002) composed of three convolutional layers followed by two fully connected layers. For FMNIST, we use a simpler architecture with two convolutional layers and a single dense layer. Local training on each client is performed using stochastic gradient descent (SGD) with a learning rate of 0.01, momentum of 0.5, weight decay of 1×10^{-4}, and a batch size of 64. Each client trains locally for 10 epochs per round. Unless stated otherwise, we run a total of 200 global communication rounds, with 20% of clients sampled per round. We report classification performance using balanced accuracy, averaged across clients to account for non-IID data distributions.

B.2 Hyperparameter Tuning

In the context FEDDAG, hyperparameters play a crucial role in determining the model's performance, stability, and robustness. To better understand the effectiveness of FEDDAG, we investigate how sensitive the algorithm is to variations in different hyperparameters.

Local Steps (t_g). The parameter t_g controls the number of local training epochs each client performs on its own data before sending gradient information to the server. This step is crucial for estimating each client's gradient direction, which is used to compute the gradient similarity matrix. Since this training is done without any federation, the resulting gradients reflect only the client's local data. The choice of t_g affects the trade-off between computation efficiency and the quality of similarity estimation. Ideally, we want t_g to be as small as possible, while still enabling the gradients to converge enough to produce meaningful similarity measurements. Table 6 shows how accuracy

varies with different values of t_g across datasets. In these experiments, the gradient similarity matrix alone (instead of combining data and gradient) was used as the proximity matrix to assess how effectively gradient information captures client similarity. The setup follows Data Distribution I (see §5) with $\alpha' = 1$, $\rho = 30\%$, and consistent hyperparameter settings. Each communication round included 10 local training steps. As shown, increasing t_g improves accuracy initially, as longer local training leads to more stable and comparable gradients. However, accuracy plateaus around $t_g = 2$ for most datasets, indicating that the gradients have sufficiently converged for reliable similarity estimation. Beyond this point, additional local steps yield diminishing returns. Therefore, $t_g = 2$ provides a good trade-off between accuracy and efficiency.

t_g	CIFAR-10	SVHN	FMNIST
1	80.81±0.59	84.82±0.24	93.18±0.11
2	83.34±0.52	90.05±0.16	93.18±0.11
3	83.34±0.52	90.05±0.16	93.18±0.11

Table 6: Test accuracy for different values of local training rounds t_g (with 10 local steps per round) across datasets, evaluated under Data Distribution I with 30% class skew and Dirichlet parameter $\alpha' = 1$.

Weight Range for Data Similarity Matrix (δ). The parameter δ controls the sensitivity of the data similarity matrix to dataset size imbalance when comparing clients. Specifically, this weighting mechanism penalizes similarity scores between clients with large differences in dataset sizes, thereby reflecting the quantity shift more accurately. The impact of these size-based penalties is governed by the value of δ: smaller values result in minimal influence, while larger values increase the penalty's effect. Each computed similarity value is reweighted and normalized into the range $[1 - \delta, 1 + \delta]$, with $\delta \in [0, 1)$, allowing the final similarity score to scale by at most a factor of two. Table 7 report test accuracy for various values of δ across multiple datasets. Since the weighting mechanism primarily addresses size disparity and quantity shift, we examine its effect under the Dirichlet concentration factor: $\alpha' = 0.25$ (severe shift). To isolate the effect of different values of δ on accuracy, we use only the data similarity matrix (instead of combining data and gradient similarity) when computing the proximity matrix for clustering. These experiments follow the Data Distribution I described in §5, using identical hyperparameters. From Table 7, we observe that higher δ values (e.g., 0.6) can lead to improved clustering and accuracy. This suggests that the weighting scheme is particularly beneficial in highly heterogeneous environments, where accounting for dataset size differences enhances similarity estimation.

δ	CIFAR-10	SVHN	FMNIST
0.2	87.51±0.18	91.74±0.08	91.79±0.08
0.4	87.51±0.18	91.74±0.08	92.21±0.09
0.6	87.95±0.13	91.91±0.13	92.21±0.09
0.8	87.95±0.13	91.91±0.13	92.21±0.09
1.0	87.95±0.13	91.91±0.13	92.21±0.09

Table 7: Accuracy metrics for various values of weight range δ under Data Distribution I (Dirichlet $\alpha' = 0.25$, 20% class skew).

Top-k values in *CC-Graph*. The parameter k determines how many top-ranked clusters are selected as knowledge sources for each target cluster in the complementarity graph H. This graph guides which clusters will supply feature representations to others during the secondary encoder training phase. For every row in the complementarity score matrix (Eq. 12), only the top-k highest scoring entries are retained to form directed edges. A smaller k limits each cluster to fewer sources, possibly reducing noise but also restricting diversity. In contrast, a larger k increases the opportunities for learning from complementary clusters but may include low-quality connections that dilute representation quality.

Table 8 shows how varying the number of source clusters k in the complementarity graph H affects the performance of FEDDAG. Experiments are conducted under Data Distribution I with Dirichlet concentration parameter $\alpha' = 1$ and 30% label skew, while keeping all other hyperparameters fixed.

We observe that performance generally improves when $k \geq 2$, benefiting from knowledge transfer across multiple relevant clusters. In some cases, increasing k beyond 2 continues to help (e.g., FMNIST), while in others (e.g., SVHN), it leads to marginal drops in accuracy. This suggests that the optimal value of k depends on the dataset characteristics and the number of clusters in the current formation. We adopt $k = 2$ as a balanced choice to ensure diversity while maintaining relevance.

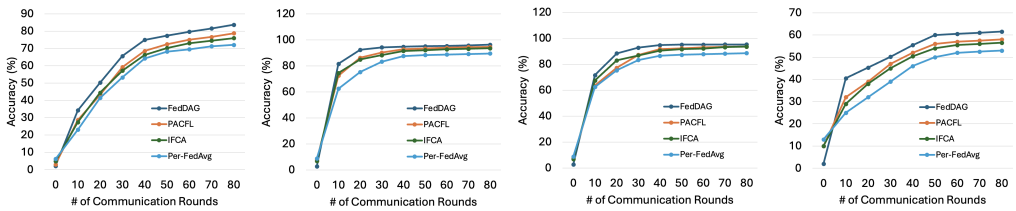
k	CIFAR-10	SVHN	FMNIST
1	90.85±0.13	97.09±0.08	97.71±0.05
2	**91.02±0.12**	**97.19±0.04**	98.36±0.10
3	90.95±0.16	97.01±0.07	**98.57±0.08**
4	90.96±0.21	96.78±0.12	98.41±0.11

Table 8: Accuracy for different values of top-k retained in the complementarity graph H under Data Distribution I (Dirichlet $\alpha' = 1$, 30% class skew).

B.3 Experiments on Data Distribution I

Exp 1: Performance Evaluation. This section presents the additional results referenced in the main paper for $\alpha' = 1$, under class skew $\rho = 20\%$ and 30%, following the setup described in §5. The results, shown in Table 9, further validate the effectiveness of FedDAG on Data Distribution I under moderate quantity shift. The same set of baselines is used, and results are reported across all four datasets.

Exp 6: Convergence Under Limited Communication rounds. We compare the performance of FedDAG against state-of-the-art (SOTA) baselines under a constrained communication budget of 80 rounds. Figure 3 reports the final local test accuracy versus the number of communication rounds for four datasets. The results demonstrate that FedDAG consistently converges within 20 to 30 communication rounds, outperforming all other methods in both convergence speed and final accuracy.



(a) CIFAR-10 (b) FMNIST (c) SVHN (d) CIFAR-100

Figure 3: Exp 6: Test accuracy versus number of communication rounds, Data Distribution I, non-IID (30%), α'=1.

B.4 Experiments on Data Distribution III

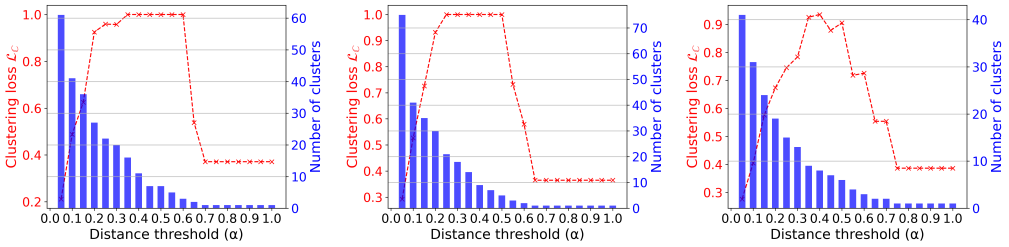
Exp 7: Finding Optimal Cluster Formation under LDA Skew. We repeat the clustering selection experiment under Data Distribution III, where client data follows an LDA-based label distribution with moderate skew ($\alpha' = 1$). Figure 4, the red curve plots clustering loss and blue bars indicate the number of clusters. Similar to Data Distribution I, the optimal threshold α^* is selected based on a balance between clustering loss and cluster count.

B.5 Data Distribution IV

This distribution evaluates FedDAG under a combination of feature skew and label skew. To simulate feature skew, we follow an approach similar to FedRC (Guo et al., 2024), which leverages datasets (e.g., CIFAR-10-C) that apply diverse image corruptions, thereby introducing different feature styles. To simulate label skew, we adopt the LDA method (Hsu et al., 2019). Specifically, each client is assigned one of the available corruption types (e.g., fog, contrast, etc. for CIFAR-10-C) to create feature skew, and the samples are distributed using the Dirichlet factor (Ng et al., 2011).

Table 9: Exp 1: Performance comparison for Data Distribution I with 20% and 30% non-IID label skew under low quantity shift (Dirichlet $\alpha' = 1$).

	20% Label Skew				30% Label Skew			
Algorithm	**CIFAR-10**	**FMNIST**	**SVHN**	**CIFAR-100**	**CIFAR-10**	**FMNIST**	**SVHN**	**CIFAR-100**
FedAvg	46.20 ± 0.97	57.12 ± 0.30	74.61 ± 0.36	51.34 ± 0.78	57.48 ± 0.17	77.17 ± 0.24	68.34 ± 0.45	53.13 ± 1.46
FedProx	46.77 ± 0.14	56.81 ± 0.16	77.23 ± 0.45	53.38 ± 0.86	57.80 ± 0.23	73.87 ± 0.25	69.65 ± 0.19	53.97 ± 0.85
PerFedAvg	84.68 ± 0.19	91.18 ± 0.21	92.34 ± 0.13	69.43 ± 0.22	82.83 ± 0.14	94.74 ± 0.17	91.48 ± 0.29	60.70 ± 0.30
FedSoft	77.42 ± 0.21	87.64 ± 0.35	90.48 ± 0.24	65.98 ± 0.37	76.94 ± 0.38	89.56 ± 0.37	84.86 ± 0.45	56.61 ± 0.31
PACFL	90.45 ± 0.30	94.41 ± 0.31	94.96 ± 0.12	70.35 ± 0.36	87.01 ± 0.38	97.28 ± 0.24	94.36 ± 0.19	63.91 ± 0.76
CFL	72.80 ± 0.66	86.97 ± 0.23	82.06 ± 0.34	61.43 ± 0.92	71.85 ± 0.79	85.67 ± 0.23	80.23 ± 0.25	52.90 ± 1.17
CFL-GP	87.83 ± 0.19	91.45 ± 0.27	90.38 ± 0.16	69.73 ± 0.20	85.67 ± 0.25	96.82 ± 0.24	92.29 ± 0.09	61.24 ± 0.73
FedGWC	89.58 ± 0.17	93.56 ± 0.09	93.67 ± 0.13	72.75 ± 0.29	86.18 ± 0.25	96.97 ± 0.14	92.94 ± 0.19	61.35 ± 0.43
FedRC	76.12 ± 0.28	86.45 ± 0.42	89.22 ± 0.31	64.78 ± 0.33	75.12 ± 0.31	91.02 ± 0.44	83.67 ± 0.38	57.89 ± 0.24
IFCA	89.68 ± 0.17	94.02 ± 0.09	93.28 ± 0.13	72.86 ± 0.29	86.42 ± 0.25	96.61 ± 0.14	92.86 ± 0.19	61.34 ± 0.43
FEDDAG	$\mathbf{94.53 \pm 0.12}$	$\mathbf{96.82 \pm 0.18}$	$\mathbf{97.04 \pm 0.23}$	$\mathbf{75.32 \pm 0.33}$	$\mathbf{91.02 \pm 0.12}$	$\mathbf{98.36 \pm 0.10}$	$\mathbf{97.19 \pm 0.04}$	$\mathbf{67.17 \pm 0.61}$



(a) CIFAR-10 (b) FMNIST (c) SVHN

Figure 4: Exp 7: Clustering score versus cluster α values and number of clusters for finding optimal clustering.

Dataset. We use two datasets for this task in the FL setting: CIFAR-10-C, TINY IMAGENET-C (Hendrycks & Dietterich, 2019).

Exp 8: Performance under Feature Skew. We evaluate the performance of SOTA algorithms and FEDDAG on different datasets under a combination of feature skew and label skew. Each client is randomly assigned one of the 20 available corruption types, and samples are distributed using a Dirichlet concentration factor $\alpha' = 1$. The results in Table 10 show that FEDDAG consistently achieves higher accuracy than the baseline methods. This improvement is attributed to FEDDAG's data-based similarity metric, which provides more accurate feature similarity estimation compared to existing approaches.

B.6 EXPERIMENT ON GENERALIZATION TO NEWCOMERS

To assess the ability of FEDDAG to generalize to unseen clients (see Appendix A.7), we simulate a dynamic federated learning environment using Data Distribution I with 30% label skew and Dirichlet concentration factor $\alpha' = 1$. Initially, training is performed on 80 out of 100 clients for 80 communication rounds, following the standard FEDDAG procedure. At the end of this phase, the remaining 20 clients join the system as newcomers. Each newcomer executes steps (1–15) of Algorithm 4 and is assigned to a cluster. Once assigned, the client receives the current global model from its designated cluster and personalizes it for 1 round (10 local epochs). To evaluate model quality, we report the average final test accuracy of the 20 newcomers across different datasets. As shown in Table 11, FEDDAG achieves better generalization to newcomers than competing methods. This improvement is attributed to its robust cluster assignment and generalization strategy for new clients.

Algorithm	CIFAR-10-C	TINY IMAGENET-C
FedAvg	30.73 ± 0.36	18.43 ± 0.43
PerFedAvg	60.39 ± 0.13	25.54 ± 0.31
PACFL	63.62 ± 0.22	33.53 ± 0.38
CFL	59.48 ± 0.15	28.97 ± 0.26
FedRC	61.82 ± 0.21	32.14 ± 0.19
IFCA	62.52 ± 0.39	32.33 ± 0.19
FEDDAG	$\mathbf{65.62 \pm 0.31}$	$\mathbf{36.27 \pm 0.32}$

Table 10: Exp 8: Performance comparison of various SOTA algorithms and FEDDAG under combined feature skew and label skew (Data Distribution IV).

Algorithm	CIFAR-10	FMNIST	SVHN
FedAvg	55.38±0.15	74.93±0.22	66.86±0.28
PerFedAvg	80.92±0.10	92.62±0.17	90.00±0.15
FedSoft	74.98±0.27	87.45±0.22	83.59±0.12
PACFL	85.33±0.15	95.17±0.24	92.76±0.08
CFL	69.97±0.11	83.64±0.13	78.94±0.17
FedGWC	84.30±0.13	94.53±0.10	91.56±0.06
FedRC	73.36±0.26	88.91±0.21	82.36±0.12
IFCA	84.55±0.22	94.61±0.30	91.58±0.22
FEDDAG	**88.23±0.18**	**96.84±0.23**	**95.74±0.13**

Table 11: Test accuracy of newcomer clients trained under Data Distribution I with 30% label skew and $\alpha' = 1$.