A Greedy Agglomerative Framework for Clustered Federated Learning

Manan Mehta and Chenhui Shao

***Abstract*—Federated learning (FL) has received widespread attention for decentralized training of deep learning models across devices while preserving privacy. Industrial Big Data in key applications such as healthcare, smart manufacturing, autonomous driving, and robotics is inherently multisource and heterogeneous. Recent studies have shown that the quality of the global FL model deteriorates in the presence of such non-IID data. To address this, we present a novel clustered FL framework called federated learning via Agglomerative Client Clustering (FLACC). FLACC greedily agglomerates clients or groups of clients based on their gradient updates while learning the global FL model. Once the clustering is complete, each cluster is separated into its own federation, allowing clients with similar underlying distributions to train together. In contrast with existing methods, FLACC does not require the number of clusters to be specified *a priori*, can handle client fractions, and is robust to hyperparameter tuning. We demonstrate the efficacy of this framework through extensive experiments on three benchmark FL datasets and an FL case study simulated using industrial mixed fault classification data. Qualitative clustering results show that FLACC accurately identifies clusters in the presence of various statistical heterogeneities in the client data. Quantitative results show that FLACC outperforms vanilla FL and state-of-the-art personalized and clustered FL methods, even when the underlying clustering structure is not apparent.**

***Index Terms*—Clustering, data heterogeneity, fault diagnosis, federated learning, Internet of Things (IoT).**

I. Introduction

THE integration of deep learning (DL)-based technologies into industrial artificial intelligence has attracted widespread attention to solve data-driven industrial problems in applications such as autonomous driving, smart grids, smart manufacturing, and healthcare, among others. Developing and training high-quality DL models requires massive amounts of data collected across different IoT devices or industrial nodes. Traditionally, a centralized entity (e.g., a cloud service provider) collects the data from all participating nodes for centralized learning. This approach raises two main concerns [1], [2], [3]. First, there is a huge communication cost in pooling data at the central server due to the scale of industrial Big Data (the number of industrial nodes and the amount of data generated at each node). Second, there are serious privacy threats as the data may contain sensitive or proprietary information.

These concerns of scalability and privacy can be addressed by using federated learning (FL), which is a decentralized framework that collaboratively learns a global model through distributed training across multiple industrial nodes [4]. The participants in FL are called *clients*, the central entity is called the *server*, and their collection is called the *data federation*. FL enables clients to learn a globally optimized model without revealing their private data to the server or to each other [5].

Albeit its popularity, recent studies have shown that the global model learned by FL may not be suitable for all clients [6]. While FL can handle non-IID data, it yields suboptimal results when the client data distributions diverge. Since industrial Big Data is inherently heterogeneous, local data distributions of individual clients are rarely identical. For instance, in healthcare applications, the distributions of the users' data vary depending on their behavioral habits and physical characteristics [7] (feature distribution skew). In quality control of 3-D printed parts using computer vision, the types of defects and their pixel intensities across different printers are different [8] (label distribution skew). In autonomous driving, the images of parked cars look different in different conditions such as day versus night and cloudy versus sunny [9] (concept drift). Finally, participating IoT devices can hold vastly different quantities of data [10] (quantity skew or unbalancedness). The impact of such statistical heterogeneity is exacerbated in FL as the server has no visibility of the clients' data.

In this article, we propose a novel client clustering framework termed as **F**ederated **L**earning via **A**gglomerative **C**lient **C**lustering (FLACC) to address this issue of divergent client data distributions in FL. We assume that the clients in the federation can be partitioned into disjoint clusters, where clients from the same clusters are likely to have the same underlying data distributions. FLACC begins with all clients belonging to their own singleton cluster. After each communication round, the server computes a similarity measure between gradient updates sent by the clients and greedily agglomerates the most similar clients or groups of clients. Once the clustering halts, each group is separated into its own federation. This allows clients with the same underlying distributions to benefit by training together and restricts negative information transfer between dissimilar clients.

Manuscript received 28 July 2022; revised 20 December 2022; accepted 24 February 2023. Date of publication 6 March 2023; date of current version 18 October 2023. Paper no. TII-22-3286. *(Corresponding author: Chenhui Shao.)*

The authors are with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: mananm2@illinois.edu; chshao@illinois.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2023.3252599.

Digital Object Identifier 10.1109/TII.2023.3252599

Our framework can be readily deployed in a realistic federation because it does not require the number of clusters to be known *a priori*, can seamlessly handle arbitrary client fractions, requires little to no hyperparameter tuning, and identifies hidden clustering structures in seemingly IID data to outperform vanilla FL.

We demonstrate the broad scope of our formulation through case studies using three common FL datasets—MNIST, CIFAR-10, and EMNIST [11], [12]. To demonstrate the applicability of FLACC to industrial IoT data, we formulate a case study to classify mixed faults in rotating machinery. We provide quantitative comparisons with state-of-the-art personalized and clustered FL methods along with qualitative client clusters identified by FLACC for different data distributions. Our results show that FLACC outperforms vanilla FL and state-of-the-art methods even when they use optimal hyperparameters, and correctly identifies client clusters for all data distributions. Additionally, we perform over 100 experiments to highlight FLACC's robustness to hyperparameter selection.

II. Related Work

A. FL and Non-IID Data

FL was proposed with the federated averaging algorithm (FedAvg), which was envisioned to work with non-IID distributions unlike previous distributed training approaches [5]. However, subsequent studies revealed the limitations of FL in the presence of diverging client distributions [9]. To counter these limitations, three main lines of research are prevalent. The first line of research focuses on improving the single global model learned by FedAvg to accommodate client-level statistical heterogeneity. Examples include sharing a small subset of data across clients [13], adding a regularization term to the local optimization to restrict divergence between global and local models [14], and optimizing for a mixture of client distributions [15]. The second line of research is pluralistic and learns individually personalized models for each client. This approach is termed as personalized federated learning and can be achieved via multitask learning [16], model regularization [17], contextualization [18], local fine tuning [19], and meta-learning [20]. The third line of research is clustered FL, which assumes that groups of clients share one data distribution. In this case, a personalized model is learned for each group rather than each individual client. Since our framework falls in this category, we review the related work for clustered FL in more detail.

B. Client Clustering in FL

The clustered FL setting allows for groups of clients to share one data distribution, thereby allowing fewer underlying distributions than clients. The objective is then to train one model for each distribution. To solve the clustered FL problem, a distance-based hierarchical clustering algorithm (FL+HC) is formulated in [21], which clusters clients using agglomerative clustering after an arbitrary number of FedAvg rounds. A multicenter aggregation mechanism is presented in [22] using stochastic expectation maximization (FeSEM), which optimally matches clients to a predetermined number of centers. A client grouping algorithm (FedGroup), which is developed in [23], performs clustering by decomposing the client gradients via singular value decomposition. A training-loss-based iterative federated clustering algorithm (IFCA) is presented in [24], where each client is greedily assigned to the cluster which yields the lowest loss on its local data. A similar hypothesis-based clustering algorithm (HypCluster) is proposed in [25] for which generalization guarantees are provided. The clustered federated learning algorithm (CFL) developed in [26] recursively splits clients into optimal bipartitions based on the cosine similarities between their gradient updates and then checks for heterogeneity in the bipartitions using gradient norms. More recently, soft clustering methods such as FedEM [27] and FedSoft [28] have been developed, which allow client data to follow a mixture of distributions.

C. Research Gap and Our Contributions

Almost all existing clustered FL algorithms such as IFCA, HypCluster, FeSEM, and FedGroup require the number of clusters to be known *a priori*. FL+HC does not require the number of clusters directly, but requires an arbitrary cutoff distance that governs the number of clusters. The performance of these algorithms is predominantly contingent upon the accuracy of the prespecified number of clusters. In real-world applications, there is little to no information available about each client's data to the server. Moreover, clients can not only be clustered using their raw data but also using more nuanced features identified during training. Thus, it is challenging to accurately determine the number of clusters *a priori*. Among existing methods, only CFL does not require the number of clusters to be prespecified. Rather, it automatically identifies k clusters after $k-1$ optimal bipartitions. However, to achieve this, CFL requires all clients to participate in each training round, which is not scalable in a realistic federation with hundreds or potentially thousands of participating clients.

The proposed FLACC framework bypasses these limitations using a greedy agglomerative strategy. Our algorithm computes the cosine similarity between the selected clients' updates and greedily merges the most similar clients or groups of clients after each communication round. We formulate stopping criteria for this merging based on inter- and intra-cluster similarities to ensure optimal client clustering. We also introduce the novel idea of system memory in our framework to deal with client fractions, which is potentially useful in other branches of FL research.

Our main contributions are summarized as follows.

1) We propose a framework for clustered FL, which does not require the number of clusters *a priori*, but rather automatically identifies optimal clusters. This improves utility in realistic industrial applications, where an underlying cluster structure is rarely known.
2) Our framework seamlessly incorporates client fractions during training.
3) Our framework is empirically seen to require minimal hyperparameter tuning, i.e., clustering performance is robust to the choice of hyperparameters.

4) We demonstrate the effectiveness of our framework, both quantitatively and qualitatively, using extensive experiments on three benchmark FL datasets and an FL case study simulated using industrial mixed fault classification data.

III. Problem Formulation

A. Preliminaries

In FL, a central server and a set of m clients form a data federation wherein the ith client has a private dataset $\{(x_i^l, y_i^l)\}_{l=1}^{n_i}$ of n_i data points. Each client has a local supervised learning model parameterized by $\theta_i \in \Theta$. The empirical risk associated with client i's finite data can be written as

$$F_i(\theta) = \frac{1}{n_i} \sum_{l=1}^{n_i} \ell\left(\theta; x_i^l, y_i^l\right) \tag{1}$$

where $\ell(\theta; x, y) : \Theta \to \mathbb{R}_{\geq 0}$ is the loss function associated with a point (x, y). In the FL setting, we assume that there exists $\theta^* \in \Theta$, which can minimize the empirical risk on all m clients simultaneously and thus solve the minimization

$$\min_{\theta \in \Theta} \sum_{i=1}^{m} \frac{n_i}{n} F_i(\theta) \tag{2}$$

where $n = \sum_{i=1}^{m} n_i$ is the total number of data points across all clients.

This minimization can be solved using the FedAvg algorithm, which involves several communication rounds between the clients and the central server [5]. In communication round t, the clients first download the most recent server model $\theta^{(t)}$. Each client then trains this model on its local data for a few epochs. The model updates are sent back to the server, which averages them to obtain the global server state $\theta^{(t+1)}$.

When clients train locally for one epoch, FedAvg can equivalently perform the global update by averaging either the gradient updates or weight updates sent by the clients

$$\begin{aligned}\theta^{(t+1)} &\leftarrow \theta^{(t)} - \eta \sum_{i=1}^{m} \frac{n_k}{n} \nabla_\theta F_i(\theta^{(t)}) \\ &= \sum_{i=1}^{m} \frac{n_k}{n} \left(\theta^{(t)} - \eta \nabla_\theta F_i(\theta^{(t)})\right) \\ &= \sum_{i=1}^{m} \frac{n_k}{n} \theta_i^{(t+1)}\end{aligned}$$

where $\nabla_\theta F_i(\theta^{(t)})$ is the gradient of client i's empirical risk computed at $\theta^{(t)}$, $\theta_i^{(t+1)}$ is client i's local weight update, and η is the learning rate. For more than one local epoch, gradient and weight averaging are no longer equivalent, and FedAvg uses weight averaging to allow more client-level computation and enhance communication efficiency. However, gradient averaging can still be used with methods like unbiased gradient aggregation and controllable meta-updating [29].

Finally, FedAvg generally only uses a small fraction $c \cdot m$ of the total clients in each communication round where client fraction $c \in (0, 1]$.

B. Clustered FL Problem

In the clustered FL setting, we consider a data federation with m clients, which can be partitioned into s disjoint clusters $\mathcal{C}_1, \ldots, \mathcal{C}_s$ based on s underlying data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_s$, such that $2 \leq s \leq m$ and $\dot{\bigcup}_{k=1}^{s} \mathcal{C}_k = [m]$, where $[m]$ is the set of integers $\{1, 2, \ldots, m\}$. A client $i \in \mathcal{C}_k$ has all its data points $(x_i^l, y_i^l) \sim \mathcal{D}_k$. The true risk for cluster $\mathcal{C}_k$ is the expected loss of the data following $\mathcal{D}_k$:

$$F^k(\theta) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}_k}\left[\ell(\theta; x, y)\right]. \tag{3}$$

Our goal in clustered FL is to minimize $F^k(\theta)\ \forall k \in [s]$. In this setting, it is easy to see that the FedAvg assumption of a single optimum θ^* will not hold and will be suboptimal for most individual clusters. Thus, we wish to find s distinct optimal solutions $\{\theta^{k,*}\}_{k=1}^{s}$ such that

$$\theta^{k,*} = \arg\min_{\theta \in \Theta} F^k(\theta), \quad k \in [s]. \tag{4}$$

Since each cluster has clients with identically distributed data, and assuming for any client $i \in \mathcal{C}_k$, the empirical risk $F_i(\theta)$ approximates the true cluster risk $F^k(\theta)$ arbitrarily well, the above minimization can be solved optimally by first assigning clients to their respective clusters and then minimizing each cluster risk separately using FedAvg. Stated differently, we want to find an optimal s-partitioning among m clients, which is the many-one-onto map

$$\mathcal{P}_s : [m] \to [s] \tag{5}$$

such that $\forall\, i, j$, we have $\mathcal{P}_s(i) = \mathcal{P}_s(j)$ iff $\mathcal{D}_i = \mathcal{D}_j$. We want to compute the optimal s-partitioning in (5) when s is also unknown, i.e., the number of clusters is not supplied to the system as a hyperparameter. This is a more realistic setting for FL as very little information about client distributions is available in practice.

Consider the FedAvg iterate $\theta^{(t)}$ at communication round t. Assuming that the gradient updates from all clients are available at round t, we can define a similarity measure between two clients i, j using the cosine similarity between their gradient updates

$$\alpha_{i,j}^{(t)} = \frac{\langle \nabla_\theta F_i(\theta^{(t)}), \nabla_\theta F_j(\theta^{(t)}) \rangle}{\|\nabla_\theta F_i(\theta^{(t)})\| \|\nabla_\theta F_j(\theta^{(t)})\|} \in [-1, 1]. \tag{6}$$

Cosine similarity computes the directional similarity between gradients and is not affected by gradient magnitudes. In FL, gradient magnitudes between clients may have disproportionate differences and adversely affect measures utilizing L1 or L2 distances [30]. As is conjectured in most clustered FL frameworks, gradient updates from clients having the same underlying distribution have higher directional similarity relative to clients having dissimilar distributions despite the batch gradients being noisy approximations of the true gradient, thereby resulting in higher values of $\alpha_{i,j}$ [21], [22], [23], [26].

The optimal s-partitioning problem at round t can then be formulated using the cosine similarities between client updates. The optimal number of clusters and the corresponding clustering structure can be obtained by minimizing the average maximum

similarity between clients belonging to different clusters

$$s^*, \mathcal{C}^* \leftarrow \underset{\substack{s\in\{2,\dots,m\}\\ \{\mathcal{C}\}_s}}{\arg\min} \frac{2}{s(s-1)} \sum_{\substack{\mathcal{C}_i,\mathcal{C}_j\in\{\mathcal{C}\}_s\\ \mathcal{C}_i\neq\mathcal{C}_j}} \max_{i\in\mathcal{C}_i j\in\mathcal{C}_j} \alpha_{i,j}^{(t)} \quad (7)$$

where $\{\mathcal{C}\}_s$ represents one of many possible ways to create the clustering map $\mathcal{P}_s$. For a fixed number of clusters s, the set $\{\mathcal{C}\}_s$ can be generated in $\binom{m}{s}$ different ways, each of which satisfies $|\{\mathcal{C}\}_s| = s$ and $\dot{\bigcup}_{k=1}^{s}\mathcal{C}_k = [m]$.

Remark 1: A brute force solution search for (7) scans all possible number of partitions $s \in \{2,\dots,m\}$ and all $\binom{m}{s}$ clusters for each s, with each evaluation computing $\max \alpha_{i,j}$ over $s(s-1)/2$ tuples. This is intractable when the number of clients m is large.

Remark 2: Equation (7) yields meaningful results only when the cosine similarity $\alpha_{i,j}^{(t)}$ between all client pairs is available at round t, which is not true in practice.

Remark 3: On fixing $s^* = 2$ and $\theta^{(t)} = \theta^*$, we obtain

$$\mathcal{C}_1^*, \mathcal{C}_2^* \leftarrow \underset{\mathcal{C}_1\dot{\cup}\mathcal{C}_2=[m]}{\arg\min} \left(\max_{i\in\mathcal{C}_1 j\in\mathcal{C}_2} \alpha_{i,j} \right)$$

which is exactly the optimal bipartitioning used in CFL [26] and can be solved in $\mathcal{O}(m^3)$. Thus, the bipartitioning in CFL can be interpreted as a special case of (7).

IV. Federated Learning Via Agglomerative Client Clustering

A. Intuition

A brute force search for s^* and $\mathcal{C}^*$ in (7) is nontractable and since only a small fraction of clients participate in any round, the cosine similarity for a majority of client pairs is not available after each round. To accommodate this, rather than minimizing the maximum similarity between clients in different clusters, we propose to greedily maximize the minimum similarity to group clients in similar clusters. We illustrate this simple agglomerative strategy below.

Consider the gradient updates of seven clients from three underlying distributions as shown in Fig. 1. At step 0, all clients are treated as individual "entities." In step 1, the minimum similarity across all entities is maximized. In this case, the two most similar clients viz. client 1 and client 2 are clustered together to form a new "subserver." For all future comparisons, this newly formed subserver is treated as one entity. Similarly, in steps 2 and 3, two new subservers are formed with clients 6, 7 and clients 4, 5, respectively, by maximizing the minimum inter-entity similarities. In step 4, the only remaining client 3 is to be added to one of the three subservers. Client 3 has the least similarity to clients 1, 7, and 5 in subservers 1, 2, and 3, respectively. We thus compute $\arg\max\{\alpha_{1,3}, \alpha_{7,3}, \alpha_{5,3}\}$ and add client 3 to subserver 1. The three subservers formed thus represent the optimal clustering solution viz. $s^* = 3$ and $\mathcal{C}^* = \{(1,2,3),(4,5),(6,7)\}$.

This idea of successively maximizing the minimum similarity across entities is central to the FLACC framework. Since only a limited number of entities are greedily agglomerated at each step, clients can be clustered "as they come" rather than clustering in a one-shot manner. This notion implicitly handles client fractions and allows clients to be grouped from the first communication round rather than after FedAvg converges to θ^* (which involves a nontrivial communication cost).

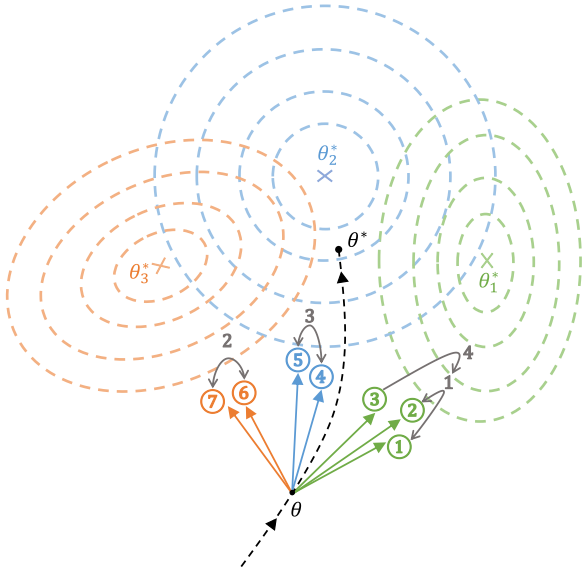

Fig. 1. Simplified schematic to illustrate the proposed greedy agglomerative strategy.

B. Algorithm

The FLACC algorithm follows the FedAvg procedure with an additional computation at the server after each communication round wherein similar clients or similar groups of clients are indexed to the same clusters. Once all clients have been assigned to clusters or no new clusters are formed for a few FedAvg rounds, each cluster is separated into its own independent federation.

Let $\theta^{(t)}$ indicate the FedAvg iterate at the beginning of round t and $M^{(t)}$ indicate the random subset of clients chosen at round t. Let $\alpha^{(t)} \in [-1,1]^{m\times m}$ be the similarity matrix whose (i,j)th element is the cosine similarity between the gradient updates of client i and client j. Let $\mathcal{E}^{(t)}$ be the entity set whose elements are either individual clients or subservers of clients, which have been agglomerated together. Both $\alpha^{(t)}$ and $\mathcal{E}^{(t)}$ contain information from the past communication rounds $0,\dots,t-1$, and are updated after every communication round.

The algorithm begins with all clients as their own singleton entities. Thus, $\mathcal{E}^{(0)}$ contains all m clients and $\alpha^{(0)} = -\mathbf{1}_{m\times m}$. In round t, after the gradient updates from all clients in $M^{(t)}$ are sent back to the server, first update the similarity matrix for all pairs of clients in $M^{(t)}$

$$\alpha_{i,j}^{(t)} = \frac{\langle \nabla_\theta F_i(\theta^{(t)}), \nabla_\theta F_j(\theta^{(t)})\rangle}{\|\nabla_\theta F_i(\theta^{(t)})\|\|\nabla_\theta F_j(\theta^{(t)})\|} \quad \forall i,j \in M^{(t)}. \quad (8)$$

We identify the pair of entities for which the minimum inter-entity similarity is maximized using (9). The two entities are

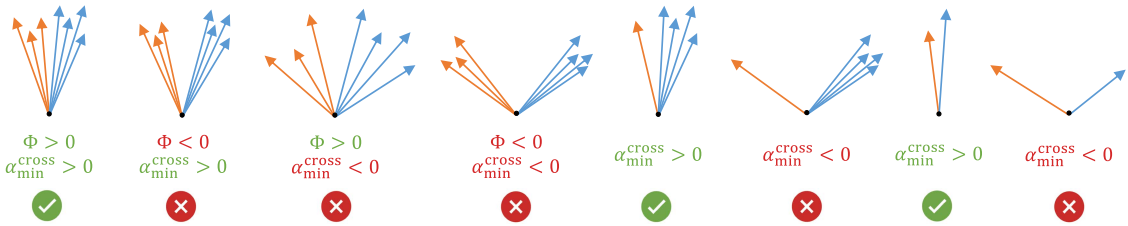

Fig. 2. Visual representation of different merging conditions for all entity pairs.

combined together to update the entity set (10)

$$E_1, E_2 \leftarrow \underset{\substack{\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}^{(t-1)} \\ \mathcal{E}_i \neq \mathcal{E}_j}}{\arg\max} \left(\min_{i \in \mathcal{E}_i, j \in \mathcal{E}_j} \alpha_{i,j}^{(t)} \right) \tag{9}$$

$$\mathcal{E}^{(t)} \leftarrow \mathcal{E}^{(t-1)} \setminus \{E_1, E_2\} \cup \{E_1 \cup E_2\}. \tag{10}$$

Based on the identities of E_1 and E_2, three types of combinations are possible.

1) If both E_1 and E_2 are clients, a new subserver is created with E_1 and E_2.
2) If E_1 is a client and E_2 is a subserver (or vice versa), client E_1 is added to subserver E_2.
3) If both E_1 and E_2 are subservers, a new subserver is created, which contains all the clients of both E_1 and E_2.

Next, we formulate two necessary conditions for entities E_1 and E_2 to be combined. In the first condition, we restrict the combination of two subservers to when their cross-subserver similarity is higher than the within-subserver similarity. We quantify this intuitive notion by defining

$$\alpha_{\max}^{\text{cross}} = \max_{i \in E_1, j \in E_2} \alpha_{i,j}^{(t)}, \tag{11}$$

$$\alpha_{\min}^{\text{within}} = \min \left(\min_{i,j \in E_1} \alpha_{i,j}^{(t)}, \min_{i,j \in E_2} \alpha_{i,j}^{(t)} \right) \tag{12}$$

and

$$\Phi(E_1, E_2) = \alpha_{\max}^{\text{cross}} - \alpha_{\min}^{\text{within}} \tag{13}$$

where $\Phi(E_1, E_2)$ is termed as the subserver combination potential. We only combine subservers E_1 and E_2 if

$$\Phi(E_1, E_2) > 0. \tag{14}$$

In the second condition, we place a lower bound on the minimum cross-entity similarity required between the entities E_1 and E_2 for them to be combined. Consider the value of $\alpha_{i,j}^{(t)}$, which solves (9)

$$\alpha_{\min}^{\text{cross}} = \min_{i \in E_1, j \in E_2} \alpha_{i,j}^{(t)}. \tag{15}$$

Then, we only combine E_1 and E_2 if

$$\alpha_{\min}^{\text{cross}} > \alpha_0 \tag{16}$$

where $\alpha_0 \in (-1, 1]$ represents the minimum allowable cross-entity cosine similarity for combination and can be set as a hyperparameter depending on the federation. In our experiments, we show that realistically, when no prior information about the federation is available, setting $\alpha_0 = 0$ is sufficient. The condition

$$\alpha_{\min}^{\text{cross}} > 0 \tag{17}$$

restricts the gradient updates of the combining entities to the "more-similar" half-plane, not allowing entities with conflicting gradients to merge. Equation (17) acts as a contingency condition preventing undesirable entity combinations in rare cases, where more similar entities are not available for merging. This notion has been used in multitask learning literature to mitigate conflicting gradients [31], [32], [33] and in FL literature to improve fairness [30]. This geometric interpretability of $\alpha_{\min}^{\text{cross}} > \alpha_0 = 0$ is another advantage of using cosine similarity as formulating a stopping criterion other distance-based similarity measures would not be equally interpretable and require careful tuning of an arbitrary constant. A visual representation of the merging conditions for all entity pairs is illustrated in Fig. 2.

At each communication round, the cosine similarities are computed between the selected client pairs, which takes $\mathcal{O}(|M^{(t)}|^2)$ time. This is followed by entity merging based on (9) and (10), subject to (17) when merging any two entities and (14) when merging two subservers. In practice, separate "entity matrices" can be utilized to store the maximum and minimum similarities between entity pairs, which are updated after each round in $\mathcal{O}(1)$ time. Using these matrices, (9) can be computed in $\mathcal{O}(|\mathcal{E}^{(t)}|^2)$ time. Overall, at each communication round, the FLACC server performs $\mathcal{O}(|M^{(t)}|^2 + |\mathcal{E}^{(t)}|^2)$ additional computation over FedAvg.

Once there is no entity merging for a predetermined number of consecutive rounds (N_{sep}), each remaining entity is separated into its own federation. FL then proceeds independently for each cluster for the remaining communication rounds.

C. Practical Considerations

In this section, we discuss some practical aspects for implementing the FLACC algorithm. When all clients participate in local training, the cosine similarity between all client pairs is updated in every round. This is not the case when only a small fraction of clients is selected in each round. Consider the two clients shown in Fig. 3 belonging to different underlying distributions, which are selected for local training in $M^{(0)}$, i.e., the first communication round. $\alpha_{i,j}^{(0)}$ for these clients is updated at round 0 and leaks through all future rounds until the two clients are selected together again. At a future round t, the true similarity between the two clients is significantly smaller than the value of $\alpha_{i,j}^{(t)}$, which may lead to incorrect agglomeration. To mitigate this, we introduce the idea of a system "memory" for

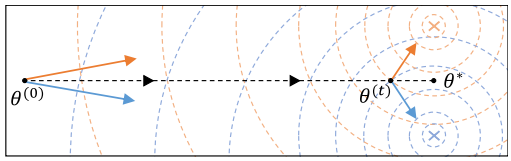

Fig. 3. Schematic to illustrate the need for a system memory in FLACC.

updating the cosine similarity matrix. Let T be the memory of the system and τ be the latest round when clients i, j appeared in training together. At round $t \geq \tau$,

$$\alpha_{i,j}^{(t)} = \begin{cases} \alpha_{i,j}^{(\tau)} & \text{if } t - \tau \leq T \\ -1 & \text{if } t - \tau > T. \end{cases} \tag{18}$$

Thus, the similarity matrix "remembers" only those client pairs, which were selected for training in the last T communication rounds, thereby preventing incorrect clustering due to similarity values leaked from older rounds. We account for this numerically in the algorithm by computing (9), (11), (12), and (15) subject to $\alpha_{i,j}^{(t)} > -1$.

FLACC computes the similarity measure between clients at each communication round using the accumulated model gradient, which is a stochastic variable. It is commonly conjectured that the accumulated gradients in distributed SGD approximate the full-batch gradient of the objective function [34]. For a sufficiently locally smooth loss function and low learning rate, a weight update over one epoch can be shown to approximate the direction of the true gradient [26]. Moreover, using weight updates for computing similarity is empirically shown to have a better signal-to-noise ratio in other clustered FL frameworks [21], [26]. We thus use weight updates $\Delta\theta^{(t)}$ instead of gradient updates $\nabla_\theta F(\theta^{(t)})$ to compute the cosine similarities between clients

$$\alpha_{i,j}^{(t)} = \frac{\left\langle \Delta\theta_i^{(t)}, \Delta\theta_j^{(t)} \right\rangle}{\|\Delta\theta_i^{(t)}\| \|\Delta\theta_j^{(t)}\|}. \tag{19}$$

Correspondingly, we use model averaging at the server instead of gradient averaging, which also better aligns with the original formulation of the FedAvg algorithm [5]. In Section VII, we show an empirical comparison between using weight updates and gradients for different data distributions.

When the number of clients is very large, the agglomeration process can be expedited in practice. Rather than merging only one entity pair after each communication round, the best $1 < n_{\text{merge}} < |M^{(t)}|/2$ pairs can be merged greedily. For federations with no information about the underlying clustering structure, using $n_{\text{merge}} = 1$ is recommended.

Finally, note that an unusually small client fraction especially in the initial communication rounds might hamper the overall clustering results despite (14) and (17). For s estimated underlying clusters in the federation, it is recommended to have at least $|M^{(t)}| = s$ clients participating in each round, with $|M^{(t)}| \geq 2\ s$ to ensure ideal qualitative clustering.

The FLACC algorithm is formally presented in Algorithm 1.

Algorithm 1: FLACC.

Input: Number of clients m, each parameterized by $\{\theta_i\}_1^m$, client fraction c, local learning rate η, local epochs E, batch size B, memory T, number of entities to merge per round n_{merge}, number of rounds before separation N_{sep}, server rounds N_{server}, server initialization $\theta^{(0)}$.
Initialize $\alpha^{(0)}, \tau^{(0)}, \mathcal{E}^{(0)} \leftarrow -\mathbf{1}_{m\times m}, \mathbf{0}_{m\times m}, [m]$
for $t = 1, \ldots, N_{\text{server}}$ **do**
  $M^{(t)} \leftarrow$ random subset of $c \cdot m$ clients
  if any entities merged in previous N_{sep} rounds **then**
    for client $i \in M^{(t)}$ **in parallel do**
      $\theta_i^{(t)} \leftarrow \texttt{ClientUpdate}(\theta^{(t-1)})$
      $\Delta\theta_i^{(t)} = \theta_i^{(t)} - \theta^{(t-1)}$
    end for
    for clients $i, j \in M^{(t)}$ **do**
      $\alpha_{i,j}^{(t)}, \tau_{i,j}^{(t)} = \frac{\langle \Delta\theta_i^{(t)}, \Delta\theta_j^{(t)} \rangle}{\|\Delta\theta_i^{(t)}\| \|\Delta\theta_j^{(t)}\|}, t$
    end for
    for $k = 1, \ldots, n_{\text{merge}}$ **do**
      $E_1, E_2 \leftarrow \underset{\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}^{(t-1)}}{\arg\max} \left(\min_{i\in\mathcal{E}_i, j\in\mathcal{E}_j} \alpha_{i,j}^{(t)}\right)$
      if $\alpha_{\min}^{\text{cross}} > 0$ **and** $\Phi > 0$ (for sub-servers) **then**
        $\mathcal{E}^{(t-1)} \leftarrow \mathcal{E}^{(t-1)} \setminus \{E_1, E_2\} \cup \{E_1 \cup E_2\}$
      end if
    end for
    $\mathcal{E}^{(t)} \leftarrow \mathcal{E}^{(t-1)}$, $\alpha^{(t)} \leftarrow -1$ where $t - \tau^{(t)} > T$
    Model averaging $\theta^{(t)} \leftarrow \sum_{i\in M^{(t)}} \frac{n_i}{\sum n_i} \theta_i^{(t)}$
  else
    for client $i \in M^{(t)}$ **in parallel do**
      $\theta_i^{(t)} \leftarrow \texttt{ClientUpdate}(\theta_{\mathcal{E}}^{(t-1)})$
    end for
    for each final entity $\mathcal{E}$ **do**
      Model averaging $\theta_{\mathcal{E}}^{(t)} \leftarrow \sum_{i\in M^{(t)}\cap\mathcal{E}} \frac{n_i}{\sum n_i} \theta_i^{(t)}$
    end for
  end if
end for
$\texttt{ClientUpdate}(\theta)$ at the ith client
for each local epoch $1, \ldots, E$ **do**
  for each mini-batch of size B **do**
    $\theta \leftarrow \theta - \eta \nabla_\theta F_i(\theta)$
  end for
end for

V. Performance Evaluation

A. Data Distributions

The performance of FLACC is examined on MNIST, CIFAR-10, EMNIST, (and FEMNIST), which have been widely used as benchmark datasets for FL and clustered FL [21], [24], [26]. We examine the following five data distributions.

Rotated MNIST: The MNIST digit recognition dataset contains 60 000 training examples of 28 × 28 grayscale images and 10 classes. We first randomly assign between [200, 800] data points to $m = 100$ clients in an IID manner. We then

apply rotations of 0°, 90°, 180°, and 270° to 10, 20, 30, and 40 clients, respectively, resulting in four underlying distributions. This federation has unbalanced data within each client and an unbalanced number of clients from each underlying distribution. Moreover, this represents a *concept drift* where the conditional distribution $P(x|y)$ diverges between clients, i.e., the same label corresponds to different features in different clients.

Grouped CIFAR-10: The CIFAR-10 dataset contains 50 000 training examples of $32 \times 32 \times 3$ images and 10 classes. We separate $m = 100$ clients uniformly into five groups such that clients from each group contain training examples with two randomly assigned labels only. For instance, all clients from group 1 have labels $\{0, 4\}$ only, from group 2 have labels $\{2, 3\}$ only, and so on. Each client is randomly assigned between [300, 600] data points. This represents a *label distribution skew*, where the marginal distribution $P(y)$ varies between clients, i.e., data labels are not evenly distributed.

Label-swapped CIFAR-10: We first randomly assign between [300, 600] data points $m = 100$ clients in an IID manner. We then separate the 100 clients uniformly into five groups. For clients in each group, two labels are randomly swapped. For instance, all clients from group 1 have labels 0 and 5 swapped, from group 2 have labels 6 and 8 swapped, and so on. This represents a *concept shift* where the conditional distribution $P(y|x)$ diverges between clients, i.e., different labels are assigned to the same feature in different clients.

Grouped + Rotated EMNIST: The EMNIST dataset extends MNIST to include lower and upper case alphabets along with digits [11]. Of these, we use the 52 alphabet classes (26 lower + 26 upper case). We first assign between $[200, 800]$ data points to $m = 100$ clients such that 50 clients contain lower case alphabets only and 50 clients contain upper case alphabets only. From both groups, we randomly select 15 clients and rotate their images by 90° resulting in a total of four underlying distributions. This federation has *concept drift* along with *label distribution skew*.

FEMNIST: The FEMNIST dataset is a federated version of EMNIST, where the data are prepartitioned based on authors of the characters [12]. A random subset of $m = 195$ clients (authors) is selected from this dataset, where each client has between $[40, 525]$ data points. Although there are no fixed underlying distributions, there might be an ambiguous cluster structure due to a *feature distribution skew* (e.g., authors having similar writing styles may be clustered). The goal here is to examine whether FLACC, by identifying hidden clusters, outperforms vanilla FL even when the data distribution among clients is very similar.

B. Experimental Setup

For each data distribution, five independent runs are performed using FLACC. Each independent run comprises the following steps: First, the data are randomly split into clients as described in the previous section. Then, each client's data are randomly partitioned into an 85%-15% train-test split. Finally, FLACC is run using a random selection of clients at each communication round. Such a shuffled selection of clients helps evaluate the robustness of the clustering algorithm. For each client, the test accuracy is evaluated on its local test data and the average accuracy over all clients is stored for each independent run. In the results section, we report the mean and standard deviation of the test accuracies over the five independent runs.

For all distributions, we use a simple CNN with two 5×5 convolutional layers with 32 and 64 channels respectively, and ReLU activations. Each convolutional layer is followed by a 2×2 max pooling layer. The output from the convolutions is passed through a fully connected layer with 512 units and ReLU activations. The network outputs a softmax classification over ten classes for MNIST and CIFAR-10, and 62 classes for EMNIST. SGD is used as the local solver with learning rate $\eta = 0.1, 0.075, 0.05$ for MNIST, CIFAR-10, and EMNIST, respectively.

Unless otherwise stated, the following parameters are used throughout all experiments: Client fraction of $c = 0.2$, i.e., 20 clients out of 100 (39 out of 195 for FEMNIST) are randomly chosen for training in each round. Local models are trained for $E = 5$ rounds with a batch size of $B = 32$. $n_{\text{merge}} = 2$ entities are merged in each round and a system memory of $T = 10$ rounds is used. Each server is trained for $N_{\text{server}} = 100$ rounds ($N_{\text{server}} = 200$ for FEMNIST) with $N_{\text{sep}} = 10$ nonmerging rounds before entity separation.

C. Results

The average test accuracy and qualitative clustering results for one independent run are illustrated in Fig. 4. In the entity merging stage, FLACC keeps track of client clusters by indexing clients and subservers at each communication round. In this stage, the test accuracies for FLACC and FedAvg are identical. Once there is no entity merging for N_{sep} consecutive rounds, the entities identified by FLACC are separated into their own federations. The single optimum θ^* learned by FedAvg is suboptimal for all clients due to the mutually competing objectives of underlying distributions. On separation, each cluster learns the optimal $\theta^{k,*}$ for its own underlying distribution. This causes a distinct and immediate jump in the client test accuracy as seen at approximately round 60 in all four cases in Fig. 4 (top).

FLACC correctly separates clients into their respective clusters in each case despite unbalanced data among clients and an unbalanced number of clients in different clusters. The clustering, as seen in Fig. 4 (bottom), is found to be exact every time for all five independent runs.

Fig. 5 shows the test accuracy comparison for the FEMNIST case. FLACC separates the 195 clients into 11 or 12 clusters depending on the initialization. There is a small yet distinct increase in test accuracy after separation at round 140, similar to Fig. 4. This result shows that FLACC outperforms FedAvg by identifying an underlying cluster structure even when client separation is not obvious.

A numerical comparison of test accuracy over five independent runs is shown in Table I. FLACC is compared to NoFed, FedAvg, two state-of-the-art personalized FL methods pFedMe [17], Per-FedAvg [20], and three state-of-the-art clustered FL methods IFCA [24], FL+HC [21], and CFL [26]. For NoFed, each client learns only from its local data, and model averaging is not performed. For IFCA, the number of

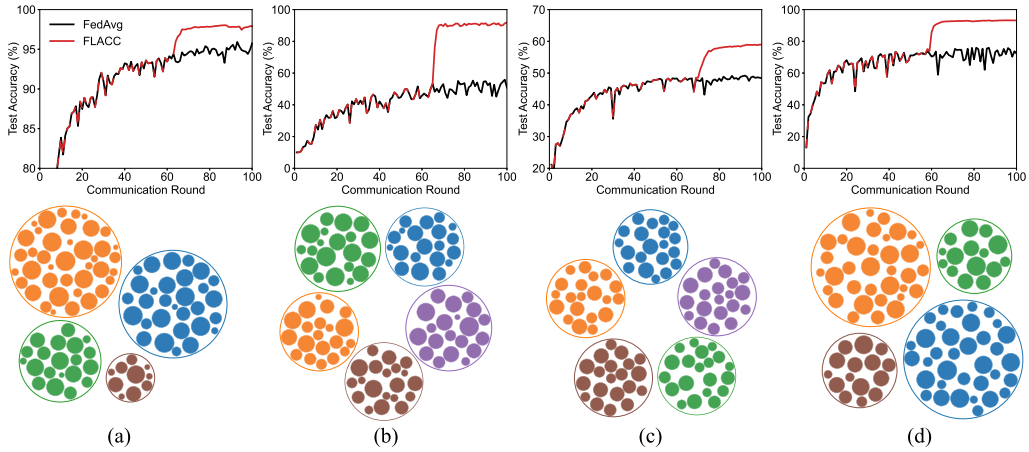

Fig. 4. Average test accuracy comparison between FedAvg and FLACC (top) and client clusters identified by FLACC (bottom) for 100 clients in (a) rotated MNIST, (b) grouped CIFAR-10, (c) label-swapped CIFAR-10, (d) grouped+rotated EMNIST. Each circle represents a client—circle color indicates the underlying distribution of the client's data and circle size represents the number of data points.

TABLE I
TEST ACCURACY COMPARISON BETWEEN FLACC, FEDAVG, AND STATE-OF-THE-ART PERSONALIZED AND CLUSTERED FL METHODS

	Rotated MNIST	Grouped CIFAR-10	Label-swapped CIFAR-10	Grouped+Rotated EMNIST	FEMNIST
NoFed	91.37 ± 0.06	80.68 ± 0.04	32.64 ± 0.31	74.69 ± 0.09	65.11 ± 0.32
FedAvg	95.30 ± 0.31	52.93 ± 2.13	48.30 ± 0.38	73.15 ± 1.49	77.57 ± 0.48
pFedMe (λ, β)	96.09 ± 0.31 (15, 4)	68.69 ± 1.02 (15, 6)	38.13 ± 2.18 (10, 8)	82.92 ± 2.98 (15, 4)	52.92 ± 4.61 (15, 1)
Per-FedAvg (α)	95.55 ± 0.57 (0.01)	64.16 ± 1.28 (0.005)	45.21 ± 2.76 (0.01)	78.69 ± 3.11 (0.015)	56.02 ± 3.80 (0.01)
CFL	97.56 ± 0.04	**89.75 ± 0.52**	55.93 ± 0.26	90.04 ± 0.18	72.42 ± 3.21
IFCA (k)	96.79 ± 0.09 (3) **98.24 ± 0.11 (4)** 97.24 ± 0.09 (5)	84.04 ± 1.36 (4) **90.17 ± 0.16 (5)** 89.38 ± 0.08 (6)	48.09 ± 0.61 (4) 48.57 ± 0.62 (5) 48.90 ± 1.18 (6)	89.86 ± 0.18 (3) 91.59 ± 0.89 (4) 91.25 ± 0.76 (5)	76.03 ± 0.14 (2) 77.64 ± 0.16 (5) 76.63 ± 0.26 (8)
FL+HC (d)	95.21 ± 0.10 (0.50) 96.46 ± 0.12 (0.75) 85.96 ± 0.13 (1.00)	81.89 ± 0.86 (0.4) 86.45 ± 0.32 (0.6) 80.67 ± 3.13 (0.9)	39.84 ± 0.70 (0.8) 45.81 ± 2.64 (0.9) 42.95 ± 0.37 (1.0)	87.36 ± 1.78 (0.60) 90.22 ± 0.13 (0.75) 79.61 ± 3.86 (0.90)	70.61 ± 1.44 (0.7) 71.85 ± 2.99 (0.8) 69.25 ± 2.85 (0.9)
FLACC	97.92 ± 0.01	**90.05 ± 0.17**	**58.30 ± 0.48**	**93.29 ± 0.11**	**82.92 ± 0.30**

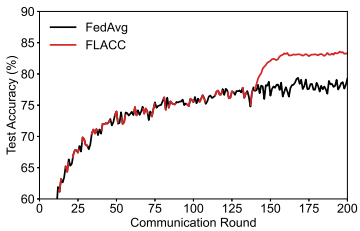

Fig. 5. Average test accuracy comparison between FedAvg and FLACC for FEMNIST.

clusters (k) and for FL+HC, the cutoff distance (d) to stop hierarchical clustering must be prespecified. Test accuracy for both methods is reported for the optimal value of the respective hyperparameters along with two other values (one higher and one lower). CFL does not require the number of clusters but requires all 100 clients to participate in each round. The regularization λ and global model update parameter β for pFedMe, and the step size α for Per-FedAvg, are carefully tuned and the results with the best parameters are reported. All other hyperparameters are kept consistent across methods.

Table I shows that vanilla FL (FedAvg) is clearly not suitable for most cases due to divergent client distributions. Moreover, FLACC matches or outperforms other methods even when those methods use optimal hyperparameters. The performance of IFCA (and FL+HC) is suboptimal when the number of clusters (and cut-off distance) is specified incorrectly. CFL sometimes matches the performance of FLACC; however, each CFL run incurs five times more communication cost than FLACC. Both personalized FL models report better performance than FedAvg, but considerably worse performance than FLACC as they are constrained in personalization by limiting deviations from a single global model. The datasets analyzed have multiple (sometimes mutually exclusive) latent contexts across local data distributions, which the personalized FL models may not capture. This has also been shown to degrade overall performance due to negative information transfer in personalized models [35]. Finally, in the FEMNIST case when the underlying distributions are not obvious, FedAvg outperforms other methods due to the

TABLE II
LABELS FOR THE 12 MIXED FAULTS

		Rotor			
		A1B1O	A1B1A	A2O	A3A
Bearing	Slight Ball Wear	0	1	2	3
	Severe Ball Wear	4	5	6	7
	Outer Race Damage	8	9	10	11

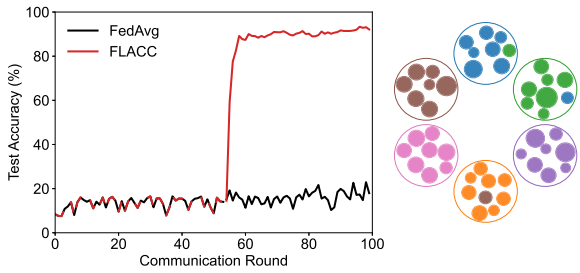

Fig. 6. Average test accuracy comparison between FedAvg and FLACC (left) and client clusters identified by FLACC (right) for mixed fault classification.

seemingly IID nature of the data. Even in this case, FLACC outperforms FedAvg and other methods by automatically finding a hidden clustering structure among clients.

VI. MIXED FAULT CLASSIFICATION

In this section, we examine the effectiveness of FLACC using a simulated case study on an industrial machine fault dataset. This dataset is collected on a specialized rotating machinery fault simulator equipped with an electric motor, two rotor disks (A and B), and a shaft supported on two bearings. Lateral vibration signals are collected using an accelerometer placed on the shaft. Controlled simulations are performed on various fault conditions associated with the rotors and the bearings. To simulate a realistic scenario, data for combinations of rotor and bearing faults, i.e., mixed faults are collected.

In particular, 4 rotor and 3 bearing conditions are simulated resulting in a total of 12 machine health conditions. Rotor conditions are simulated by attaching additional standard weights to the rotor disks A and B. The four rotor conditions are: one weight on A, one weight diametrically opposite on B (A1B1O); one weight on A, one weight on B aligned (A1B1A); two weights on A diametrically opposite (A2O); three adjacent weights on A (A3A). The three bearing conditions are: slight ball wear, severe ball wear, and outer race damage. The labels for the 12 conditions are shown in Table II.

Each fault label in Table II has 1920 data points (signals) resulting in 23 040 data points in total. We form a federation with $m = 48$ clients separated into six groups with eight clients per group. Clients from groups 1 to 6 have fault labels (0, 1), (2, 3), (4, 8), (5, 6), (7, 11), and (9, 10), respectively. Each client is randomly assigned between [200, 600] signals from its respective group labels. Such client grouping is done to emulate a realistic federation, where all clients do not have all fault types. Moreover, there is feature overlap between clients from different groups. For instance, clients from groups 1 and 2 both have slight ball wear and can only be distinguished based on their rotor faults. Similarly, clients from each group have a feature overlap with clients from (at least) four other groups. This makes the mixed fault classification task using FL very challenging.

We construct a 1-D CNN having three 1-D convolutional layers with ReLU activations, each followed by a max pooling layer. The convolutional layers have 64, 128, and 256 filters, respectively, of size 5. The output from the convolutions is passed through two fully connected layers (128 and 64 units, respectively) with ReLU activations. The output is a softmax classification over the 12 fault classes. Adam is used as the local solver with $\eta = 5\text{e}{-}04$. We use $c = 0.25$, $E = 5$, $B = 32$, $n_{\text{merge}} = 1$, $T = 10$, $N_{\text{server}} = 100$, and $N_{\text{sep}} = 10$.

Fig. 6 shows the test accuracy comparison between FedAvg and FLACC along with the clusters identified by FLACC. Vanilla FL is not well suited for the realistic client grouping in this case, as shown by the FedAvg accuracy which stagnates at $\sim$20%. Contrarily, FLACC identifies and clusters similar entities based on their fault signals and separates them into independent federations, leading to a 4.5 times increase in the test accuracy to $\sim$90%. Unlike the qualitative results in Fig. 4, the clustering, in this case, is not exact. While the number of identified clusters is always correct, two or four clients are incorrectly clustered in each trial depending on the random initialization. However, the realistic data distribution in this case is also more challenging than in previous studies.

Table III shows the numerical comparison between test accuracy for FLACC and other state-of-the-art methods over five independent runs. FLACC outperforms other methods even when optimal hyperparameters are used for the other methods. Overall, this case study proves the usefulness of FLACC in complex real-world industrial applications.

VII. DISCUSSION

A. Effect of Hyperparameters

In this section, we analyze the effect of two main hyperparameters used in the FLACC framework—the minimum allowable cross-entity cosine similarity for merging two entities (α_0) and the system memory (T)—and empirically demonstrate that FLACC requires minimal hyperparameter tuning.

1) Effect of α_0: Experiments for rotated MNIST, grouped CIFAR-10, label-swapped CIFAR-10, and grouped+rotated EMNIST are run using three values of α_0 viz. $+0.1, 0, -0.1$, keeping all other parameters identical to Section V-B. The test accuracy and the number of clusters identified by FLACC in five independent runs are shown in Table IV.

When $\alpha_0 = 0$, FLACC identifies the underlying clusters perfectly in all five independent runs for all four datasets. When $\alpha_0 = +0.1$, FLACC identifies more than the actual underlying number of clusters, whereas, when $\alpha_0 = -0.1$, fewer clusters are identified. This is expected, as higher values of α_0 place stricter similarity requirements resulting in fewer merges and

TABLE III
TEST ACCURACY COMPARISON BETWEEN FLACC AND STATE-OF-THE-ART METHODS FOR MIXED FAULT CLASSIFICATION

NoFed	FedAvg	pFedMe (λ, β)	Per-FedAvg (α)	CFL	IFCA (k)	FL+HC (d)	FLACC
85.17±1.40	21.56±1.43	42.92±4.08 (30, 5)	26.36±2.73 (0.005)	87.54±0.8	66.89±2.32 (4) 89.98±1.74 (6) 87.72±1.71 (8)	62.11±2.23 (0.2) 65.36±1.33 (0.5) 24.21±0.85 (0.8)	**92.21±0.70**

TABLE IV
TEST ACCURACY AND NUMBER OF CLUSTERS FOR DIFFERENT MINIMUM ALLOWABLE CROSS-ENTITY COSINE SIMILARITIES (α_0)

	$\alpha_0 = +0.1$		$\alpha_0 = 0$		$\alpha_0 = -0.1$	
	Test Accuracy	N Clusters	Test Accuracy	N Clusters	Test Accuracy	N Clusters
Rotated MNIST	97.47 ± 0.05	12.80 ± 0.98	97.92 ± 0.01	4 ± 0	96.75 ± 0.31	2 ± 0
Grouped CIFAR-10	89.86 ± 0.13	5 ± 0	90.05 ± 0.17	5 ± 0	87.63 ± 3.91	4.80 ± 0.40
Label-swapped CIFAR-10	54.70 ± 4.26	14.20 ± 4.02	58.30 ± 0.48	5 ± 0	49.94 ± 0.34	1 ± 0
Grouped+Rotated EMNIST	92.20 ± 0.21	10.2 ± 1.47	93.29 ± 0.11	4 ± 0	83.26 ± 2.54	2 ± 0

TABLE V
TEST ACCURACY AND NUMBER OF CLUSTERS FOR DIFFERENT SYSTEM MEMORIES (T)

	$T = 2$		$T = 5$		$T = 10$		$T = 25$		$T = \infty$	
	Test Accuracy	N Clusters	Test Accuracy	N Clusters	Test Accuracy	N Clusters	Test Accuracy	N Clusters	Test Accuracy	N Clusters
Rotated MNIST	97.53±0.15	3.8±0.4	97.85±0.06	4±0	97.92±0.01	4±0	97.93±0.09	4±0	97.54±0.23	6.80±0.98
Grouped CIFAR-10	89.94±0.24	5±0	89.17±0.14	5±0	90.05±0.17	5±0	90.09±0.15	5±0	89.67±0.21	5±0
Label-swapped CIFAR-10	58.17±0.61	5±0	58.62±0.33	5±0	58.30±0.48	5±0	57.89±0.62	5±0	53.19±1.97	10.60±2.65
Grouped+ Rotated EMNIST	93.24±0.18	3.8±0.4	92.84±0.19	4±0	93.29±0.11	4±0	93.25±0.07	4±0	91.89±0.58	4±1.10

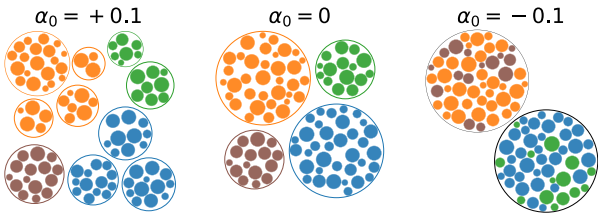

Fig. 7. Representative clusters for Grouped + Rotated EMNIST using different minimum allowable cross-entity cosine similarities.

more clusters. Lower values of α_0 place loose similarity requirements resulting in more merges and fewer clusters. This idea is illustrated using representative clusters identified for Grouped + Rotated EMNIST in Fig. 7.

Contrary to the number of clusters, the test accuracy is affected in one direction only. For $\alpha_0 = +0.1$, despite more clusters being identified, the clients in each cluster are still similar. Thus, on separation, each group learns a good model for their underlying data and the test accuracy is consistent with $\alpha_0 = 0$ case. However, for $\alpha_0 = -0.1$, clusters contain dissimilar clients which deteriorate performance.

We emphasize that $\alpha_0 = 0$ yields consistently accurate results across all four data distributions. While α_0 does implicitly control the number of clusters identified by FLACC, unlike existing methods, α_0 need not be tuned differently for different data distributions.

2) Effect of T: Keeping all other parameters unchanged, experiments are run for different system memories T chosen across a broad range. When $T = 2$, the system has a very short-term memory and has access to the cosine similarities from the last two communication rounds only. When $T = \infty$, the system has access to the similarities from all previous rounds, which is equivalent to not using a memory formulation. The results for five independent runs are shown in Table V.

The test accuracy and number of clusters identified by FLACC remain optimal and nearly consistent across a broad range of T. However, when $T = \infty$, the number of clusters is no longer optimal due to cosine similarities computed in early communication rounds causing incorrect merges in latter rounds. While the test accuracy using $T = \infty$ is not severely affected, the variability in the number of identified clusters may lead to incorrect inferences regarding the true underlying clustering structure.

These experiments show that the use of a system memory T ensures consistent and optimal results across different data

TABLE VI
TEST ACCURACY AND NUMBER OF CLUSTERS USING WEIGHT UPDATES VERSUS GRADIENTS FOR COSINE SIMILARITY

	Weight Updates		Gradients	
	Test Accuracy	N Clusters	Test Accuracy	N Clusters
Rotated MNIST	97.92±0.01	4±0	97.69±0.02	4±0
Grouped CIFAR-10	90.05±0.17	5±0	90.18±0.15	5±0
Label-swapped CIFAR-10	58.30±0.48	5±0	52.40±2.52	5.2±1.16
Grouped+ Rotated EMNIST	93.29±0.11	4±0	90.15±0.36	4±0.4

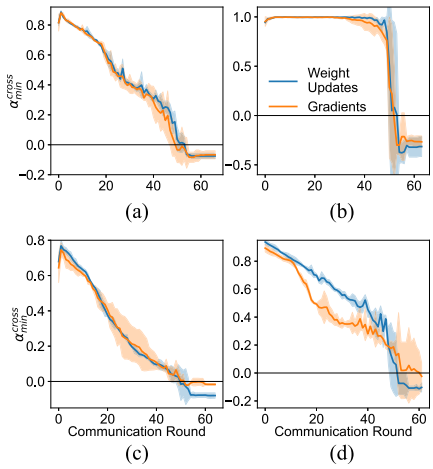

Fig. 8. Value of $\alpha_{\min}^{\text{cross}}$ which solves (9) at each communication round using weight updates versus gradients for cosine similarity. (a) Rotated MNIST (b) Grouped CIFAR-10. (c) Label-swapped CIFAR-10. (d) Grouped+Rotated EMNIST.

distributions; however, careful tuning of T is not necessary to ensure such results.

B. Weight Updates Versus Gradients

In this section, we compare the performance of FLACC when using gradients from the individual client models in order to compute the cosine similarity to the results obtained in Section V using weight updates. The test accuracy and number of clusters obtained are shown in Table VI. Using gradients instead of weight updates does not hamper the clustering performance for Rotated MNIST and Grouped CIFAR-10; however, leads to occasional incorrect clustering in Label-swapped CIFAR-10 and Grouped+Rotated EMNIST.

Both the gradients and weight updates are stochastic in nature; thus, the similarity measure built upon them is a noisy one. The signal required from these noisy measures is a meaningful value of $\alpha_{\min}^{\text{cross}}$, which solves (9) to yield the entities to merge after each round. Fig. 8 plots the values of $\alpha_{\min}^{\text{cross}}$ when using weight updates and gradients for the four data distributions. $\alpha_{\min}^{\text{cross}}$ decreases with communication rounds, which is expected owing to the greedy nature of the algorithm. While the values of $\alpha_{\min}^{\text{cross}}$ are relatively consistent between the weight updates and gradients, the variability in the gradient-based cosine similarity is higher across all four distributions. This higher noise leads to the occasional incorrect clustering in the Label-swapped CIFAR-10 and Grouped+Rotated EMNIST distributions.

C. Limitations and Future Work

The FLACC formulation is inherently stochastic and relies on extracting useful signals from noisy SGD updates. While we show compelling empirical results, a theoretical analysis to study the interplay between client selection, SGD updates, hyperparameters, and the corresponding clustering results would be insightful. Sharing gradient or weight updates directly with the central server compromises the privacy of individual clients. In practice, privacy mechanisms are always utilized with FL to protect individual clients' data. An important next step is exploring the utility of FLACC with privacy mechanisms like differential privacy and gradient encryption. We study the clustered FL problem assuming that clients come from disjoint generative distributions. A more general formulation for a mixture of generative distributions is an interesting direction for future research.

VIII. CONCLUSION

The performance of FL deteriorates when the client data distributions are non-IID. This issue is commonly encountered in numerous industrial applications, where client data are inherently heterogeneous, thereby critically limiting the successful deployment of FL. This article develops FLACC—a novel framework for clustering similar clients in FL. FLACC agglomerates clients or groups of clients while learning the global FL model and then separates each cluster to be trained independently. Case studies on widely used FL datasets show that FLACC outperforms FedAvg and state-of-the-art clustered FL methods. Moreover, using a mixed fault diagnosis dataset, we demonstrate the efficacy of FLACC in more complex and realistic industrial federations. Our framework is readily applicable to industrial settings because it is autonomous (automatically identifies the optimal clustering structure) and scalable (incorporates client fractions during training).

REFERENCES

[1] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial IoT," *IEEE Trans. Ind. Inform.*, vol. 17, no. 8, pp. 5709–5718, Aug. 2021.
[2] Q. Han, S. Yang, X. Ren, P. Zhao, C. Zhao, and Y. Wang, "PCFed: Privacy-enhanced and communication-efficient federated learning for industrial IoTs," *IEEE Trans. Ind. Inform.*, vol. 18, no. 9, pp. 6181–6191, Sep. 2022.
[3] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6532–6542, Oct. 2020.
[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.
[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Art. Intell. Statist.*, 2017, pp. 1273–1282.

[6] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
[7] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Inform. Res.*, vol. 5, no. 1, pp. 1–19, 2021.
[8] M. Mehta and C. Shao, "Federated learning-based semantic segmentation for pixel-wise defect detection in additive manufacturing," *J. Manuf. Syst.*, vol. 64, pp. 197–210, 2022.
[9] P. Kairouz et al., "Advances and open problems in federated learning," *Foundations Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.
[10] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.
[11] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *Proc. IEEE Int. Joint Conf. Neural Netw*, 2017, pp. 2921–2926.
[12] S. Caldas et al., "LEAF: A benchmark for federated settings," 2018, *arXiv:1812.01097*.
[13] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.
[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.
[15] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proc. Int. Conf. Mach. Learn*, 2019, pp. 4615–4625.
[16] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 4424–4434.
[17] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with Moreau envelopes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 21394–21405.
[18] P. P. Liang et al., "Think locally, act globally: Federated learning with local and global representations," in *Proc. Workshop Feder. Learn. NeurIPS*, 2019, pp. 1–34.
[19] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.
[20] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 3557–3568.
[21] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *Proc. Int. Jt Conf. Neural Netw*, 2020, pp. 1–9.
[22] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning: Clients clustering for better personalization," *World Wide Web*, vol. 26, no. 1, pp. 481–500, 2023.
[23] M. Duan et al., "FedGroup: Efficient clustered federated learning via decomposed data-driven measure," in *Proc. IEEE 19th Int. Symp. Parallel Distrib. Process. Appl.*, 2021, pp. 228–237.
[24] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 19586–19597.
[25] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, *arXiv:2002.10619*.
[26] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural. Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3710–3722, Aug. 2021.
[27] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 15434–15447.
[28] Y. Ruan and C. Joe-Wong, "Fedsoft: Soft clustered federated learning with proximal local updating," in *Proc. Conf. AAAI Artif. Intell.*, 2022, vol. 36, pp. 8124–8131.
[29] X. Yao, T. Huang, R.-X. Zhang, R. Li, and L. Sun, "Federated learning with unbiased gradient aggregation and controllable meta updating," in *Proc. Workshop Federated Learn. Data Privacy Confidentiality*, 2019, pp. 1–16.
[30] Z. Wang, X. Fan, J. Qi, C. Wen, C. Wang, and R. Yu, "Federated learning with fair averaging," in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 1615–1623.
[31] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 5824–5836.
[32] Z. Chenet al., "Just pick a sign: Optimizing deep multitask models with gradient sign dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 2039–2050.
[33] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 18878–18890.
[34] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.
[35] X. Tang, S. Guo, and J. Guo, "Personalized federated learning with contextualized generalization," in *Proc. Int. Joint Conf. Artif. Intell. Org.*, 2022, pp. 2241–2247.

Manan Mehta received the B.E. (Hons.) degree in mechanical engineering from Birla Institute of Technology and Science, Pilani, India, in 2019. He is currently working toward the Ph.D. degree in mechanical engineering with a graduate minor in statistics at the University of Illinois at Urbana-Champaign, Champaign, IL, USA.

His research focuses on developing data-efficient statistical and machine learning techniques for applications in smart manufacturing and Internet of Things. His particular interests include federated learning, Gaussian processes, multitask learning, and sequential decision-making.

Chenhui Shao received the B.E. degree in automation from the University of Science and Technology of China, Hefei, Anhui, China, in 2009; the M.S.E. degree in industrial and operations engineering, the M.A. degree in statistics, and the Ph.D. degree in mechanical engineering, all from the University of Michigan, Ann Arbor, MI, USA, in 2013, 2013, and 2016, respectively.

He is an Associate Professor with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign. His research interests focus on smart manufacturing, machine learning, statistics, materials joining, and manufacturing systems control and automation.

Dr. Shao's honors and awards include NSF CAREER Award, ASME Chao and Trigger Young Manufacturing Engineer Award, IISE Manufacturing and Design Division Outstanding Young Investigator Award, SME Barbara M. Fossum Outstanding Young Manufacturing Engineer Award, SME 30 Under 30 Honoree, and Dean's Award for Excellence in Research at Illinois. His research contributions have been recognized by multiple best paper awards and the Manufacturing USA Report.