Group-based Client Sampling in Multi-Model Federated Learning

Zejun Gong*
Electrical and Computer Engineering
Carnegie Mellon University
Pittburgh, PA, USA
zejung@alumni.cmu.edu

Haoran Zhang*
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
haoranz5@andrew.cmu.edu

Marie Siew
Information Systems Technology and Design Pillar
Singapore University of Technology and Design
Singapore
marie_siew@sutd.edu.sg

Carlee Joe-Wong
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
cjoewong@andrew.cmu.edu

Rachid El-Azouzi
CERI/LIA
University of Avignon
Avignon, France
rachid.elazouzi@univ-avignon.fr

***Abstract*—Federated learning (FL) allows multiple clients to collaboratively train a model without sharing their private data. In practical scenarios, clients frequently engage in training multiple models concurrently, referred to as multi-model federated learning (MMFL). While concurrent training is generally faster than training one model at a time, MMFL exacerbates traditional FL challenges like the presence of non-i.i.d. data: since each individual client may only be able to train one model in each training round due to local resource limitations, the set of clients training each model will change in each round, introducing instability when clients have different data distributions. Existing single-model FL approaches leverage inherent client clustering to accelerate convergence in the presence of such data heterogeneity. However, since each MMFL model may train on a different dataset, extending these ideas to MMFL requires creating a unified cluster or group structure that supports all models while coordinating their training. In this paper, we present the first group-based client-model allocation scheme in MMFL able to accelerate the training process and improve MMFL performance. We also consider a more realistic scenario in which models and clients can dynamically join the system during training. Empirical studies in real-world datasets show that our MMFL algorithms outperform several baselines up to 15%, particularly in more complex and statically heterogeneous scenarios.**

***Index Terms*—Federated Learning, Resource Allocation, Multi-Model Federated Learning**

I. Introduction

In Federated Learning (FL), edge devices collaboratively train a shared model locally without sharing their private data [1]. The typical FL setting assumes that one single model is collectively trained. However, in many real-world scenarios, there is a need for Multi-Model Federated Learning (MMFL) [2]–[6], where multiple models are trained concurrently across the same set of clients. For example, intelligent vehicles generate diverse and sensitive data that can be used for the simultaneous training of multiple models, such as those for driving-related tasks (e.g., localization, mapping, and control), safety-related tasks (e.g., driver monitoring and environmental analysis), and infotainment systems. MMFL allows these models to be updated concurrently, maintaining performance within required time frames and addressing privacy concerns more effectively than traditional centralized methods. Compared to sequentially training one model at a time, concurrent training has been shown to accelerate model convergence [2], [3], [7].

Single-model FL algorithms aim to leverage a diverse group of devices to collaboratively train a single global model. These algorithms focus on making the best use of the computational power of each client and local data to ensure the success of this centralized training task. Numerous existing FL algorithms have been proposed in the literature, each seeking to optimize this process by addressing challenges such as model aggregation [8], and client heterogeneity [9]. To extend these to MMFL settings, we follow prior works [2]–[4], [6] in assuming that each client can only train a single model in any given round. In vehicular networks, while certain high-end vehicles possess the computational and storage capacity to handle multiple tasks per round, enforcing this on a broader vehicular fleet overlooks their diversity and limitations. Prioritizing more capable vehicles risks over-utilization, potentially compromising their primary functions, while sidelining less capable ones disregards their valuable data and raises fairness concerns. More importantly, this assumption aligns well with the inherent constraints of vehicular networks: autonomous vehicles can generate up to one gigabyte of data per second [10], creating significant demands on onboard processing and communication. Combined with high mobility and intermittent connectivity, training multiple models simultaneously would amplify transmission overhead and synchronization complexity, increasing the risk of delays and corrupted updates. Limiting individual clients' tasks in each round minimizes the communication overhead and synchronization challenges that

This work is supported by NSF CNS-2106891 and ANR-22-CE23-0024, and the SUTD-MOE Faculty Early Career Award.

*Contributed equally to this work.

can arise from simultaneously broadcasting and aggregating the parameters of multiple models. Thus, we maintain a scalable framework for multi-model FL deployments, striking a balance between resource constraints and the need to capture the diverse data distributions across all participating clients. However, each model consequently experiences **partial client participation** due to heterogeneous communication and computation resource constraints, in every round, which is known to de-stabilize training even in the single-model FL case if not managed carefully [11], [12].

Prior MMFL works generally do not optimize the set of clients selected to train each model [2], [3], [5], leading to a high variance in the client updates toward each global model for each task. When data distributions vary significantly between clients, ignoring these disparities leads to slower convergence and suboptimal performance. Indeed, many works have focused on reducing this global variance in single-model FL [11], [13], [14]. These works mitigate the impact of high variance in partial participation by adjusting server aggregation rules to balance client participation [13], [15], introducing additional information (e.g., stale updates) to stabilize updates [9], [11], [14], [16], or leveraging data distributions to form client groups [12], [17] and then selecting only clients from a given group in each training round.

While some of these methods can be easily extended to MMFL settings, *client grouping* cannot, as we explain below. Thus, in this paper, our objective is to devise a client grouping strategy for an MMFL system that reduces the variance of model updates, speeding up convergence while respecting the constraints of each client to train only one model in each training round.

One of the most common ways of dividing clients into groups in the single-model FL setting is to cluster clients according to the characteristics of their local data. Previous work on client clustering groups clients based on dataset size [18] or gradient similarity [17], [19], with different clustering algorithms (bipartitioning [19], soft clustering [18] and K-means [17]).

However, these approaches are fundamentally tied to single-task settings and struggle to generalize to MMFL, leaving several key challenges unaddressed, including: 1) The optimal clustering of clients for each model is likely different, e.g., due to variations in task complexity or data similarity. For instance, tasks with highly correlated data distributions could benefit from sharing the same cluster structure, whereas tasks with significantly different data may need entirely separate clusters. However, since each client can only train a single model in each training round, this cluster inconsistency makes it fundamentally impossible to simply apply the optimal client clustering for each model: we need to select a single model for each client to train.

Consequently, we need to seek a *unified* clustering structure that maximizes heterogeneity across all client groups—ensuring that, wherever possible, the same clients remain grouped together across tasks (i.e., models), even when tasks' similarity can differ substantially.

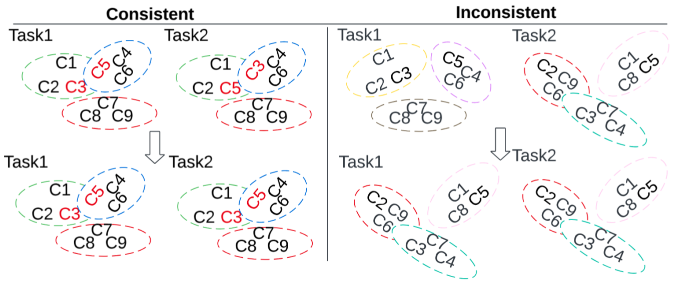


Fig. 1: Overview of the process of achieving unified cluster structure. Each oval represents a cluster within a task. ($\{C_i\}_{i=1}^{9}$) represents the client within each cluster.

Given a unified client grouping, we can assign each group of clients to one model for training, rotating these assignments in a round robin manner to ensure that all clients train each model. However, the number of client groups is often determined by the natural clustering structure, while the number of training models is set by the system, which means they may not always match. To avoid leaving groups idle and consequently wasting their training resources, it is essential to carefully manage the rotation between models and groups in our round-robin setting. Figure 1 illustrates the process of achieving a unified clustering structure for 2 tasks, each internally has 3 clusters. The consistent case indicates that task 1 and task 2 have a high correlated data distribution, while in the inconsistent scenario, task 1 and task 2 have very distinct cluster structure. Our goal is to eventually ensure that the same clients are grouped together across tasks, as indicated in the figure.

These challenges are further exacerbated by the potential **dynamics** of the clients and training models present in the system. For example, newly deployed vehicles may introduce previously unseen driving environments or sensor configurations into the MMFL system. Likewise, an automotive manufacturer may add new training tasks, such as road hazard detection, alongside existing objectives like lane keeping or traffic sign recognition. Managing the arrival of new clients has been studied in single-model FL [20], but in MMFL, there is no prior work discussing the arrival of new models.

This dual challenge requires MMFL to adapt to varying client participation while dynamically allocating resources across an expanding set of models. The non-IID nature of real-world data, coupled with new clients and models, can disrupt resource allocation, worsen data heterogeneity, and complicate model-group rotation across rounds. Thus, strategies that dynamically adjust to new clients and models are critical for the robustness and scalability of MMFL systems. To our knowledge, *we are the first to address client and model (task) dynamics in MMFL.*

Our Contributions are summarized as follows:

- We propose an effective grouping framework for MMFL, which groups clients according to their data distributions. Our framework handles scenarios where MMFL models

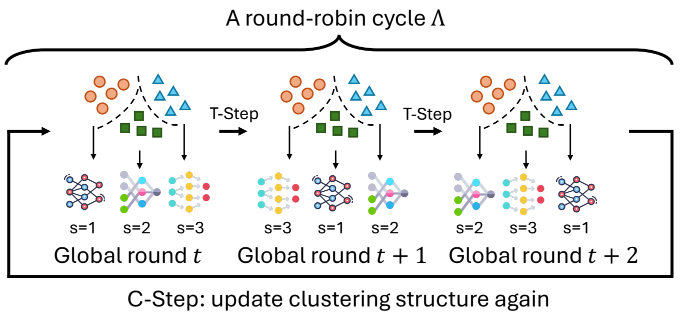


Fig. 2: Illustration of the proposed algorithm with $S = 3$ models. In each round-robin cycle Λ, clients are divided into L_Λ groups (determined as described in Section II-A1). During a global round t in the cycle Λ, each model $s = 1, 2, 3$ select clients exclusively from different groups ($\{\mathcal{C}_{\Lambda,g}\}_{g=1}^{L_\Lambda}$), represented as circles (orange), triangles (blue), and squares (green). Each model is trained by every group in a round-robin manner. For example, t: ($s = 1 \to$ group $\mathcal{C}_{\Lambda,1}$, $s = 2 \to \mathcal{C}_{\Lambda,2}$, $s = 3 \to \mathcal{C}_{\Lambda,3}$), $t + 1$: ($s = 1 \to$ group $\mathcal{C}_{\Lambda,3}$, $s = 2 \to \mathcal{C}_{\Lambda,1}$, $s = 3 \to \mathcal{C}_{\Lambda,2}$), and $t + 2$: ($s = 1 \to$ group $\mathcal{C}_{\Lambda,2}$, $s = 2 \to \mathcal{C}_{\Lambda,3}$, $s = 3 \to \mathcal{C}_{\Lambda,1}$).
After each cycle, the clustering structure is updated. Notice that we restrict $L_\Lambda \geq S$. The figure illustrates the case where $L_\Lambda = S$. When $L_\Lambda > S$, pseudo-models are introduced to fill up this routine, as detailed in Section II-A1 T-Step.

either share a similar cluster structure or have overlapping group structures across models.

- Given client groups (clusters), we propose a round-robin rotation scheme to match training models to each groups exclusively, accelerating the convergence for all models.
- We study the dynamic settings, in which new clients or models (i.e., new tasks) can be integrated in the MMFL system, thus improving scalability by adapting our algorithm to dynamic settings.
- We conduct extensive experiments under various model-specific cluster structures with differing complexities in forming a unified structure. Our results demonstrate consistently better accuracy with improvements of up to 15% over the baselines.

In Section II, we describe our proposed grouping and round-robin training algorithms, as well as their extensions to a dynamic MMFL setting with new client and model arrivals. We experimentally evaluate these algorithms in Section III before concluding in Section IV.

II. Proposed Methods

Consider an MMFL system with S models and N clients and our objective is to minimize the total loss on all models:

$$\min_{\theta_1,\cdots,\theta_S} L = \min_{\theta_1,\cdots,\theta_S} \sum_{s=1}^{S}\sum_{i=1}^{N} d_{i,s} f_{i,s}(\theta_s) \tag{1}$$

where θ_s denotes the parameters of model s and $f_{i,s}(\theta_s)$ is the local objective for client i, model s, defined as the expected local loss of client's i's data distribution on model s: $f_{i,s}(\theta_s) = \frac{1}{n_{i,s}}\sum_{\xi\in\mathcal{D}_{i,s}} l(\theta_s, \xi)$, where ξ is a data sample and $\mathcal{D}_{i,s}$ denote the set of data points available to client i for model s, with $n_{i,s} = |\mathcal{D}_{i,s}|$ representing the number of data points from client i. The loss function l measures the performance for each individual data point, e.g., cross entropy, and $d_{i,s} = \frac{n_{i,s}}{\sum_{j=1}^{N} n_{j,s}}$ quantifies the proportion of client i's data relative to the total data available for model s. The total loss is a summation over the N client's loss for model s, followed by an outer summation over the loss of the S models.

Let $\mathcal{N}_t$ be the set of *clients* in round t, and $\mathcal{S}_t$ as the set of models in round t. Figure 2 provides an overview of the proposed method with $S = 3$ models and $L_\Lambda = 3$ groups. We require that each model samples clients exclusively from one cluster per global round to accelerate convergence. To efficiently manage the training of all models, we ensure that a model is exclusively trained by one cluster and shifted to another cluster in the next round, following a round-robin scheme. We index the round-robin cycle as $\Lambda = 1, 2, \ldots, T$, where each cycle Λ consists of L_Λ steps (denoted as $\tau = 1, \ldots, L_\Lambda$), where each step corresponds to a global round of federated training, indexed by $t = \sum_{\Lambda'=1}^{\Lambda-1} L_{\Lambda'} + \tau$. Note that L_Λ is equivalent to the number of clusters we generate at the beginning of a round-robin cycle. When $L_\Lambda > S$, pseudo-models are introduced to fill up the round-robin routine, which we will elaborate on in Section II-A1 (T-Step).

In each round-robin cycle Λ, we first determine the clustering structure (referred to as the **C-Step**) and then employ the round-robin rotation to conduct training over L_Λ global rounds for all models (referred to as the **T-Step**), which will be detailed in the following section. In this paper, we restrict our study to the case where the number of clusters is greater than or equal to the number of models.

A. Clustering and Round-Robin Rotation in MMFL

1) Consistent Cluster Structure Across Models: We first consider the scenario in which models have similar underlying cluster structures. For example, in vehicular networks, multiple perception or control models may exhibit similar grouping patterns based on shared attributes such as vehicle type, sensor configuration, or driving region (e.g., urban taxis, highway trucks, or rural service vehicles). **C-Step**: At the beginning of a round-robin cycle Λ (at global round $t = \sum_{\Lambda'=1}^{\Lambda-1} L_{\Lambda'} + 1$), the server distributes global models θ_s^t ($s \in \mathcal{S}_t$) to each client. Each client performs a forward pass for each model to obtain a vector $\delta_i^\Lambda = [f_{i,1}, \ldots, f_{i,s}, \ldots]$ containing the training loss of each model, and propagates this vector to the central server. At the server, these vectors are utilized as features representing the loss patterns across all models to form clusters $\{\mathcal{C}_{\Lambda,g}\}_{g=1}^{L_\Lambda}$. The server minimizes the within-cluster variance by leveraging the K-means [21] algorithm. Specifically, clients are assigned to clusters by minimizing the Euclidean distance between their loss vectors δ_i^Λ and cluster centroids $\mathbf{c}_g$: $d(\delta_i^\Lambda, \mathbf{c}_g) = \sqrt{\sum_{s\in\mathcal{S}_t}(f_{i,s} - \mathbf{c}_g[s])^2}$. Centroids are then updated as the mean of all assigned vectors:

$\mathbf{c}_g = \frac{1}{|\mathcal{C}_{\Lambda,g}|}\sum_{\delta_i^\Lambda \in \mathcal{C}_{\Lambda,g}} \delta_i^\Lambda$. We set the optimal number of clusters be at least the number of models and use the silhouette method to determine the optimal number of clusters. The clusters correspond to the client groups.

T-Step: After clustering, all clients within a cluster are assigned to train the same model. Each model is trained by a uniformly random sample of clients within its assigned cluster(s). Every such active client performs K local epochs of training on the assigned training task and then propagates the parameters to the server for aggregation. After this, the cluster is assigned to the next model for training, in a round-robin manner. When the model-group rotation finishes all L_Λ circle steps, the round-robin cycle Λ ends. Note that C-Step might create more clusters than the number of models, leading to idle clusters without a model to train in a round-robin routine. To fully utilize clusters per round, we create pseudo-models as copies of the models with the largest losses in the current stage to fill up the routine. In this case, models with higher loss may receive updates from more than one cluster. The algorithm 1 shows the pseudocode for the entire process.

2) Inconsistent Cluster Structure Across Models: We now consider the more common scenario where *models have different underlying cluster structures*, i.e., a client is likely to have different cluster neighbors across models. For example, a connected vehicle may contribute data to both a pedestrian detection model and a traffic signal prediction model, which rely on distinct sensor modalities and contextual information—thus exhibiting little correlation in their clustering behavior. In this case, the C-Step above may result in nearly random clustering outcomes: the set of clients grouped together for one task might not be grouped together for another task. This inconsistency not only complicates client-task coordination but also imposes a significant computational burden in assigning clients to one task while ensuring that clients for other tasks remain exclusive. Therefore, we propose a **modified C-Step** to again maintain a unified cluster structure for the T-step of training. In the modified C-Step, we determine the global clustering structure solely based on one model s during the current round-robin cycle Λ. In essence, the cluster structure for task s is being copied to the rest of tasks, ensuring that all tasks share a consistent set of client groups during the current round-robin cycle Λ. In this scheme, we specify the model task s also in a "round-robin" fashion across training cycles. Specifically, at the beginning of each round-robin cycle Λ, a single model $s \in \mathcal{S}t$ is selected as the clustering anchor. For instance, if task $s = 1$ defines the clustering for Λ, then all models $s' \in \mathcal{S}_t$ are trained using the same client groupings defined by task 1. In the subsequent cycle $\Lambda + 1$, we advance to the next model $s + 1$ as the new anchor, repeating the process. This unified clustering approach eliminates the need to compute separate clustering structures for each task, significantly reducing computational overhead. In the subsequent cycle $\Lambda + 1$, the global clustering structure is determined based only on model $s + 1$.

In practice, when the data distributions of different models are clearly correlated, the original C-Step can effectively produce a unified cluster structure. Otherwise, the modified C-Step is a safe choice to ensure increased inter-cluster heterogeneity for at least one model. This scenario serves as a deliberately challenging case where task distributions are entirely non-overlapping, a setting that, while extreme, helps highlight the robustness and general applicability of our unified clustering approach. While exploring alternative clustering algorithms such as GMM [22] and spectral clustering [23] may accelerate convergence under partial task overlap, detailed clustering algorithmic comparisons are not the primary focus of this work. Nonetheless, to further support the robustness of our approach, we include experiments in later sections evaluating the impact of different clustering methods on stability and accuracy. Furthermore, the modified C-step ensures fairness by allowing each task to alternately determine the global clustering structure during the round-robin cycle. This prevents any single task from disproportionately influencing the process and ensures that all tasks have an equal opportunity to influence client group assignments, promoting balanced learning outcomes and better representation across tasks.

B. Dynamic MMFL with New Clients and Models

1) Group allocation for new clients: When new clients arrive during an ongoing round-robin cycle Λ, they must be integrated into the existing clustering structure without disrupting the training progress. In contrast to the standard **C-Step**, where all clients compute loss vectors and are clustered jointly via K-means, newly arrived clients follow a lightweight assignment process. Specifically, each client i computes a local loss vector δ_i^Λ by evaluating each global model θ_s^t, and compares it to the current cluster centroids $\{\mathbf{c}_g\}_{g=1}^{L_\Lambda}$ using cosine similarity. The client is then assigned to the cluster $g^* = \arg\max_g \cos\_\mathrm{sim}(\delta_i^\Lambda, \mathbf{c}_g)$. We acknowledge that assigning clients based on initial loss vectors may introduce transient bias, especially if their local models have not yet converged or personalized. However, this approximation serves as a low-overhead, initialization that enables immediate participation without disrupting the global training loop. More importantly, the impact of such bias diminishes over time, as the C-Step is periodically re-executed every L_Λ global rounds, allowing for cluster realignment as client behavior stabilizes and models evolve. This approach ensures that the introduction of new clients does not trigger an immediate re-execution of the C-Step, which would be computationally expensive and time-consuming. Moreover, repeating the C-Step frequently, especially when new clients arrive at a high rate, would significantly delay the start of the T-Step, hindering the overall training efficiency. Note that the arrival of new clients can alter the number of clusters, as they may introduce new data distributions, which can occur in the next round-robin cycle (C-Step).

2) Arrival of new models: We also consider an MMFL system that can dynamically accommodate the arrival of new FL models. When a new model S_{new} joins the system during an ongoing cycle Λ, it should not interfere with the current training schedule. Instead, S_{new} remains inactive until the cycle completes. At the beginning of the next round-robin cycle $\Lambda+1$, the global model set is updated to $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{S_{\text{new}}\}$, and the C-Step is re-executed using the expanded set of models. Each client then computes a new loss vector $\delta_i^{\Lambda+1}$ over the updated $\mathcal{S}_{t+1}$, allowing the clustering structure to incorporate the new model's impact on client affinities. This ensures that S_{new} is fairly integrated into the current training cycle without disrupting the rotation of cluster consistency.

Algorithm 1 MMFL-Group Sampling

```
1: for Round-robin cycle $\Lambda = 1, 2, \dots, T$ do
2:     Perform C-Step according to each scenario to determine global clusters $\{\mathcal{C}_{\Lambda,g}\}$.
3:     Determine cycle length $L_\Lambda$ as the number of clusters.
4:     for cycle step $\tau = 1, 2, \dots, L_\Lambda$ do
5:         for model $s = 1, \dots, L_\Lambda$ in parallel do
6:             decides assigned cluster $g_s = (s+\tau) \bmod L$
7:             model $s$ samples clients from $\mathcal{C}_{\Lambda,g_s}$
8:         end for
9:         for sampled client $i$ in parallel do
10:            Current global round $t = \sum_{\Lambda'=1}^{\Lambda-1} L_{\Lambda'} + \tau$
11:            $\theta_{i,s}^{t+1} \leftarrow$ Local Update($\theta_{i,s}^t$) for assigned model
12:            Send update to the server.
13:        end for
14:        Server aggregates: $\theta_s^{t+1} = \frac{1}{|\mathcal{A}_{t,s}|} \sum_{i \in \mathcal{A}_{t,s}} \theta_{i,s}^t$,
15:        where $\mathcal{A}_{t,s}$ is the set of participating clients.
16:    end for
17: end for
```

III. Experiments

A. Experiment Setup

We conducted experiments using four datasets: Fashion-MNIST, CIFAR-10, MNIST, and EMNIST, forming five models (training tasks) with repeated EMNIST. To model a scenario where clients' data distributions exhibit a naturally clustered pattern, we assume 5 ground-truth clusters for each model, with each cluster containing 20% of total labels for each model. For instance, the EMNIST dataset has 47 labels, which results in each cluster's clients containing data with approximately 9-10 labels. For most experiments, we construct 30 clients, each with 30-40 data samples per model.The number of local training epochs is set to $E = 5$ for all models. We set the active participation rate for each group to be 1/6 to further simulate a heterogeneous environment. For the Fashion-MNIST and EMNIST training tasks, we construct two similar CNNs, each with two convolutional layers, two pooling layers, and two linear layers, and with different output layer sizes. For the CIFAR-10 and MNIST task, we use a pre-activation ResNet [24]. All experiments were performed for five random seeds and the average was taken.

B. Evaluation

We compare our algorithm with two **baseline methods**: *random* allocation and the MMFL round-robin algorithm [5].

TABLE I: Clustering stability and accuracy under varying task overlap levels.

Overlap %	**K-Means**	**GMM**	**Spectral**
	(Stable, Accuracy)	**(Stable, Accuracy)**	**(Stable, Accuracy)**
0%	(**45**, 54.9%)	(56, 53.4%)	(46, **55.1%**)
50%	(39, 57.8%)	(**30**, 59.3%)	(35, **60.5%**)
100%	(**28**, **61.4%**)	(30, 60.2%)	(31, 59.8%)

All algorithms adopt the same client participation rate: 1/6. Compared to our method, the round-robin algorithm randomly divides clients into 5 groups without any clustering. Both the random allocation and round-robin algorithm first select 1/6 of total clients and then assigns a model to each client.

1) Static MMFL system: We first present the results of the consistent cluster structure scenario (Section II-A1) in Figure 4a. In this scenario, MMFL-Group Sampling approach demonstrated a significant accuracy gap (>10%) compared to the other two algorithms. This improvement is likely due to the C-Step's ability to capture the cluster information well. Therefore, clients are highly homogeneous within the cluster and heterogeneous across clusters, leading to much faster convergence based on [12]'s theoretical analysis. In Fig 3, we show the individual tasks' accuracy in the first 500 training epochs which demonstrate our algorithm's good performance on each individual tasks. We provide visualizations in Figure 5 to illustrate the cluster struc ture generated by our algorithm in global rounds $t = 5, 25$. PCA is applied to reduce the dimensionality of the loss matrix used for clustering. We can observe that at round $t = 25$ the cluster structure displayed a more distinct pattern compared to round $t = 5$. These results suggest that our clustering approach gradually increases the inter-cluster (group) heterogeneity as the training goes, which can potentially enhance the convergence speed [12].

We further conducted an experiment with completely independent model cluster structures across tasks, to challenge our algorithm in the worst case. In this case, a unified cluster structure for all models is very complex or may even not exist. Applying original C-Step leads to an almost random cluster structure which would have similar result as the round-robin algorithm. Therefore, we adopt the modified C-Step as discussed in Section II-A2. Our experimental setup is as follows: each task is partitioned into five ground truth groups based on their data labels. For each task, we randomly select clients and assign them data points corresponding to a specific cluster. This process is repeated independently for each task, resulting in distinct and independent cluster structures across tasks. As shown in Figure 4b, the results illustrated a noticeable drop in accuracy compared to the outcomes in Figure 4a. This decline can be attributed to the increased complexity of the cluster structures, causing the client-task assignment for the remaining tasks to appear less representative of their respective data distributions. However, as shown in Figure 4b, the MMFL-Group Sampling method continues to exhibit strong performance with at least 6% increase compared to the baselines in accuracy among all models. To further evaluate the effectiveness of the unified clustering approach, we include

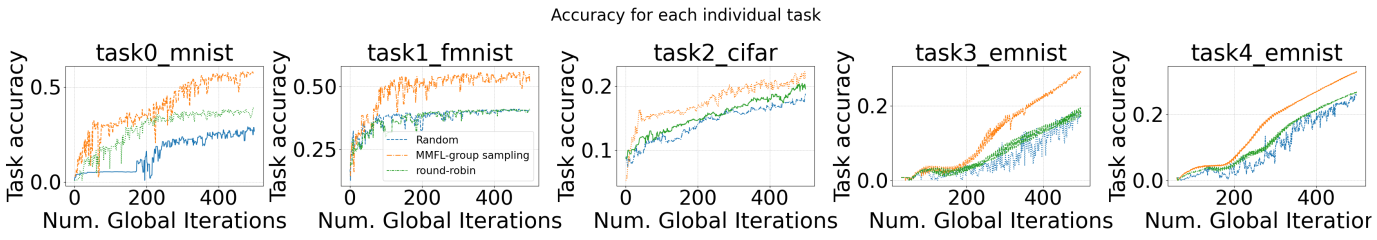
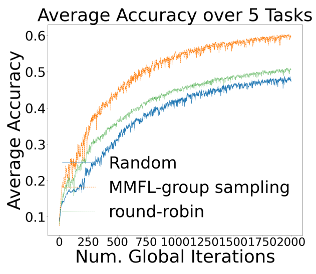
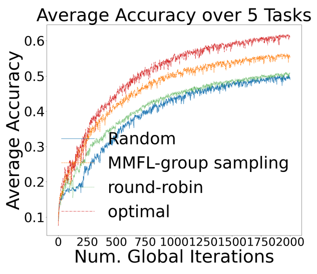


Fig. 3: Individual task's accuracy in consistent cluster structure

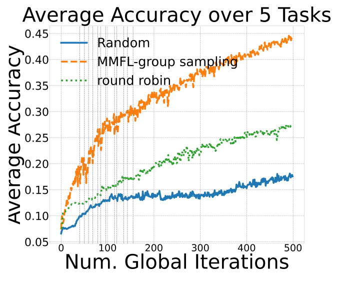


(a) Consistent Cluster Structure

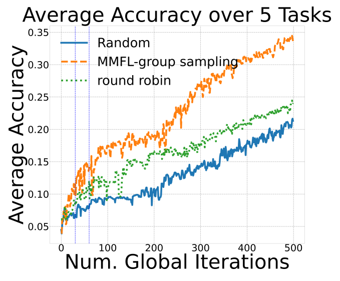


(b) Inconsistent Cluster Structure

Fig. 4: Test accuracy results of the Consistent Cluster Structure scenario from section 2.1.1 and the Inconsistent Cluster Structure scenario from 2.1.2. Our algorithm outperforms the Random Allocation and Round-robin Allocation baselines.

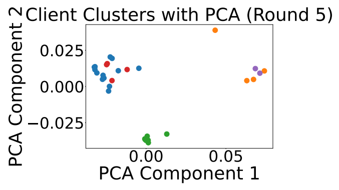


(a) Dynamic Arrival of Clients

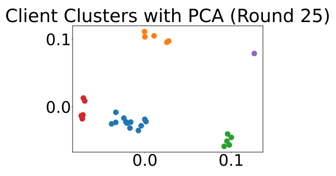


(b) Dynamic Arrival of Tasks

Fig. 6: Test accuracy for the dynamic cases. Case (a) contains 24 static clients and 36 new clients. Case (b) contains 60 clients in total.



(a)



(b)

Fig. 5: For the inconsistent cluster structure scenario, we reduce the Loss matrices dimension to 2 dimensions to illustrate the cluster structure. Here the selected task is MNIST and the clustering is more distinct at round 25 compared to round 5

a pseudo-optimal baseline in which each task maintains its own clustering structure and performs round-robin training independently. This removes the constraint of a unified cluster structure and allows each model to train on clients grouped by task-specific affinity. Although this method yields slightly higher performance due to improved task-wise specialization, it comes at the cost of increased computation: Clients may participate in multiple tasks within the same round, therefore breaking our MMFL assumption (therefore pseudo optimal). Notably, the final accuracy remains comparable to our consistent case, as both approaches benefit from leveraging the full client population with low intra-cluster variance across tasks.

To further investigate the impact of inter-task data overlap on clustering performance, we incorporate two additional clustering algorithms—GMM and spectral clustering—both of which are better suited for handling overlapping clusters. To simulate varying degrees of clustering structure overlap across tasks, we employ a controlled client assignment strategy. Given a target overlap ratio $O \in [0, 1]$, each cluster in Task 2 is formed by copying $O \times 100\%$ of its clients from the corresponding cluster in Task 1, while the remaining $(1 - O) \times 100\%$ are randomly sampled from the rest of the client pool. This design enables precise control over the consistency between task-specific cluster structures, ranging from fully independent (0% overlap) to fully aligned (100% overlap). As observed from the results in Table I, GMM and spectral clustering show advantages primarily in the moderate overlap setting (50%), where their cluster structures stabilize earlier compared to K-means. In both the low and high overlap scenarios, all three algorithms achieve comparable accuracy and convergence speed, with K-means slightly outperforming the others in some cases. These findings support our claim that while clustering algorithms designed for overlapping data can offer marginal benefits in specific conditions, the overall performance is largely attributed to our round-robin training scheme coupled with task-aware clustering.

2) Dynamic MMFL System: In Figure 6a, we present the results of new clients joining midway through the training process. The experiment involves 60 clients in total, with 24 static clients and 36 dynamic clients joining progressively. We assume that the inter-arrival time of clients follows an exponential distribution with rate λ. We set $\lambda = 0.1$ (on average 1 client joins in every 10 rounds) and the client arrival times are indicated by the dashed vertical lines in Figure 6a. Compared to Figure 4a, which features only 30 clients, the performance of random (˜27% in dynamic and ˜35% in static) and round-robin (˜17% in dynamic and ˜28% in static) methods remains similar or worsens at round 500 despite having additional clients. Conversely, MMFL-Group sampling shows improved performance (˜45% in dynamic and ˜42% in static) with the extra resources. The improved performance of the algorithm can be attributed to the systematic clustering of new clients into existing groups, which preserves the intra-group homogeneity contributions. Compared to the baselines, this approach mitigates the impact of diverse data distributions from new arrivals, ensuring more consistent updates and enhanced overall model convergence.

In Figure 6b, we present the scenario where dynamic models join during training, beginning with MNIST, Fashion MNIST, and CIFAR training task from round $t = 0$ to round $t = 30$. The first EMNIST training task is introduced in round $t = 30$, followed by the arrival of the second EMNIST training task in round $t = 60$, as marked by the dashed vertical lines. Since the experiment setup is very similar with the consistent case expect for the dynamic tasks, we observe similar average performance except for average degrade of accuracy due to the late join of dynamic tasks. Given the dynamic task's good performance, we show that with a unified cluster structure to increase inter-group heterogeneity, MMFL-Group sampling approach speeds up the convergence for new models, leading to much better performance overall. We provide more detailed experiment results in the Technical Report [25].

IV. Conclusion

In this work, we present the MMFL-Group Sampling algorithm which addresses the data heterogeneity issue in the Multi-Model Federated Learning (MMFL) system. We incorporate a loss-based grouping mechanism to group clients together, combined with a modified round-robin method which allocates models to each group. The algorithm is suitable for a practical MMFL system with static clients and tasks as well as dynamic scenarios involving the arrivals of new clients and models.Experimental results also show that our algorithm outperforms the baselines in handling cases with both consistent and inconsistent group structures. We also show an advantage in improving the scalability of the system when new clients and models join midway.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*. PMLR, 2017, pp. 1273–1282.

[2] N. Bhuyan and S. Moharir, "Multi-model federated learning," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2022, pp. 779–783.

[3] M. Siew, H. Zhang, J.-I. Park, Y. Liu, Y. Ruan, L. Su, S. Ioannidis, E. Yeh, and C. Joe-Wong, "Fair concurrent training of multiple models in federated learning," *arXiv preprint arXiv:2404.13841*, 2024.

[4] H. Zhang, Z. Li, Z. Gong, M. Siew, C. Joe-Wong, and R. El-Azouzi, "Poster: Optimal variance-reduced client sampling for multiple models federated learning," *ICDCS*, 2024.

[5] N. Bhuyan, S. Moharir, and G. Joshi, "Multi-model federated learning with provable guarantees," 2022. [Online]. Available: https://arxiv.org/abs/2207.04330

[6] M. Siew, S. Arunasalam, Y. Ruan, Z. Zhu, L. Su, S. Ioannidis, E. Yeh, and C. Joe-Wong, "Fair training of multiple federated learning models on resource constrained network devices," in *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks*, 2023, pp. 330–331.

[7] B. Askin, P. Sharma, C. Joe-Wong, and G. Joshi, "Fedast: Federated asynchronous simultaneous training," *arXiv preprint arXiv:2406.00302*.

[8] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfa, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, and Z. Sun, "Advances and open problems in federated learning," 2021. [Online]. Available: https://arxiv.org/abs/1912.04977

[9] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," 2021. [Online]. Available: https://arxiv.org/abs/1910.06378

[10] A. M. Elbir, B. Soner, S. Çöleri, D. Gündüz, and M. Bennis, "Federated learning in vehicular networks," in *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE, 2022, pp. 72–77.

[11] D. Jhunjhunwala, P. Sharma, A. Nagarkatti, and G. Joshi, "Fedvarp: Tackling the variance due to partial client participation in federated learning," in *Uncertainty in Artificial Intelligence*. PMLR, 2022.

[12] Y. J. Cho, P. Sharma, G. Joshi, Z. Xu, S. Kale, and T. Zhang, "On the convergence of federated averaging with cyclic client participation," in *International Conference on Machine Learning*. PMLR, 2023.

[13] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," *arXiv:2010.13723*, 2020.

[14] A. Rodio and G. Neglia, "Fedstale: leveraging stale client updates in federated learning," *arXiv preprint arXiv:2405.04171*, 2024.

[15] L. Wang, Y. Guo, T. Lin, and X. Tang, "Delta: Diverse client sampling for fasting federated learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[16] X. Gu, K. Huang, J. Zhang, and L. Huang, "Fast federated learning in the presence of arbitrary device unavailability," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 052–12 064, 2021.

[17] D. Song, G. Shen, D. Gao, L. Yang, X. Zhou, S. Pan, W. Lou, and F. Zhou, "Fast heterogeneous federated learning with hybrid client selection," in *Uncertainty in Artificial Intelligence*. PMLR, 2023.

[18] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "Clustered sampling: Low-variance and improved representativity for clients selection in federated learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 3407–3416.

[19] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[20] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 3403–3411.

[21] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press.

[22] D. A. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, S. Z. Li and A. K. Jain, Eds. Springer, 2009, pp. 659–663.

[23] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 14, 2001, pp. 849–856.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016, pp. 630–645.

[25] "Technical report." [Online]. Available: https://tinyurl.com/vtc-mmfl