Group-based Client Sampling in Multi-Model Federated Learning

Zejun Gong*†, Haoran Zhang*†, Marie Siew‡, Carlee Joe-Wong†, Rachid El-Azouzi§,
†Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA
‡Information Systems Technology and Design Pillar, Singapore University of Technology and Design, 487372 Singapore
§CERI/LIA, University of Avignon, Avignon 84029 France
zejung@alumni.cmu.edu, {haoranz5,cjoewong}@andrew.cmu.edu, marie_siew@sutd.edu.sg, 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). MMFL exacerbates traditional FL challenges like the presence of non-i.i.d. data: since each client may only be able to train one model in each training round, 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 the MMFL performance. We also consider a more realistic scenario in which models and clients can dynamically join the system during training. Empirical studies on real-world datasets show our MMFL algorithms outperform several baselines up to 15%, particularly in more complex and statically heterogeneous scenarios.**

***Index Terms*—Federated learning, Multi-Model Federated learning, Dynamic Resource Allocation**

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], [3], [4], where multiple models are trained concurrently across the same set of clients. For example, FL applications such as Google keyboard prediction [5], keyword-spotting [6], speech recognition [7] may each require timely updates to reflect evolving user behavior, especially under scenarios with limited time frames, such as daily usage patterns or adapting to new language inputs. In these scenarios, MMFL enables concurrent updates across multiple models, allowing them to be trained within the required time frame, thereby maintaining performance more effectively than sequentially training separate individual models.

The single-model FL assumes that all clients collectively contribute to training a single model. However, directly extending this approach to the multi-model scenario is not feasible, as clients typically lack the computational resources to train every model in each round. Therefore, in this paper, together with prior works [2], [3], [4], we assume that clients are only able to train one model in each round and, hence, each model experiences partial client participation in a single training round. However, these earlier works still suffer from high variance across all models with partial client participation since they ignore heterogeneity in client data, resulting in slow convergence and highlighting the need for an effective strategy for assigning clients to models.

There have been many works to focus on global variance reduction in single-model FL [8], [9], [10]. These works mitigate the impact of high variance in partial participation by adjusting server aggregation rules to balance client participation [8], [11], introducing extra information (e.g., stale updates) to stabilize updates [9], [10], [12], [13], or leveraging data distributions to form client groups [14], [15]. In this paper, we aim to form client groups in an MMFL system that efficiently organizes models' training for different groups in a round-robin manner, addressing the variance reduction and speeding up convergence through high inter-group heterogeneity as demonstrated for a single FL model in [15].

One of the most common ways of dividing clients into groups is via client clustering according to their local data. Previous work on client clustering groups clients based on dataset size [16] or gradient similarity [17], [14], with different clustering algorithms (bipartitioning [17], soft clustering [16] and K-means [14]). **However, directly implementing single-model FL clustering methods to form clusters in MMFL has many challenges to address:** 1) In single-model FL, client clustering typically involves forming distinct clusters tailored to a single model. Similarly in MMFL, clusters could be formed independently for each model. We aim to develop a unified clustering structure that maximizes heterogeneity across all groups. However, the varying clustering requirements of each model pose significant challenges to establishing such a unified structure, as each model may have distinct grouping needs that are difficult to reconcile within a single framework. 2) The number of 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. In this case, it is essential to carefully manage the rotation between models and groups to avoid leaving groups idle, which represents a waste of resources.

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.

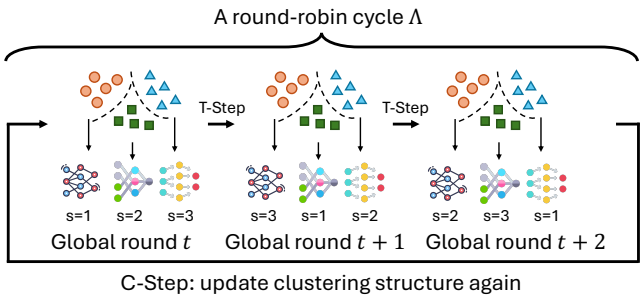


Fig. 1: Illustration of the proposed algorithm with $S = 3$ models. In each global round, models (denoted as $s = 1, 2, 3$) select clients exclusively from different clusters, represented by circles (orange), triangles (blue), and squares (green). Each model is trained by every cluster in a round-robin manner. After each cycle, the clustering structure is updated again.

These challenges are further exacerbated by the potential **dynamics** of the clients and training models present in the system. For example, *new users* of mobile devices may introduce new data distributions into the MMFL system. Similarly, a health technology company may introduce *new training tasks*, such as stress detection, alongside existing objectives such as heart rate monitoring. Managing the arrival of new clients has been studied in single-model FL [18], 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 either share a similar cluster structure or have overlapping group structures across models.
- Given client groups (clusters), we propose a 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.

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. 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 1 provides an overview of the proposed method with $S = 3$ models and $L_\Lambda = 3$ groups. Similar to the approach in [15] for single-FL, 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 consists of L_Λ steps (denoted as $\tau = 1, \ldots, L_\Lambda$), with each step corresponding to a global round $t = \sum_{\Lambda'=1}^{\Lambda-1} L_{\Lambda'} + \tau$. Note that L_Λ depends on the number of clusters we generate at the beginning of a round-robin cycle. 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, and the opposite case has been omitted for lack of space.

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 various human activity recognition or health monitoring models (using data from wearable), there may be common clusters between models, based on characteristics such as age, occupation and lifestyle (athletes vs. elderly vs. children).

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 [19] 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.

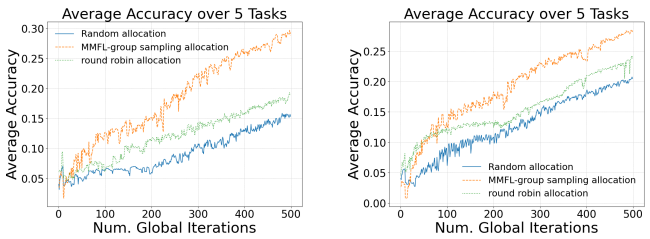
T-Step: After clustering, all clients within a cluster are assigned to train the same model. Each model is trained by a uniformly random sample of clients within its assigned cluster(s). Every such active client performs K local epochs of training on the assigned training task and then propagates the 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. Algorithm 1 shows the pseudocode for the whole 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 neighbours across models. For example, a smartphone user can generate data for speech recognition as well as active calorie data for health monitoring, which are intuitively unlikely to be correlated. In this case, the C-Step above may result in nearly random clustering outcomes. To address the global clustering issue in such cases, we propose a modified C-Step. 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 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 — for instance, when multiple models utilize similar or identical datasets (such as in speech recognition and keyword spotting) — 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.

B. Dynamic MMFL with New Clients and Models

1) Group allocation for new clients: When new clients arrive, the server requests their local loss $f_{i,s}$ for each model. We maintain the current clustering pattern and compute the cosine similarity of the new clients' loss vectors with the existing cluster centroids $\{c_g\}_{g=1}^{L_\Lambda}$ to decide which cluster they belong to. Then we perform the normal training routine. 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 consider an MMFL system that can accept new FL models. In this case, the newly joined model waits until the current round-robin cycle finishes to performs the **C-step** along with the other models and performs the training.



(a) Consistent Cluster Structure (b) Inconsistent Cluster Structure

Fig. 2: Test accuracy of the consistent structure in section 2.1.1 and inconsistent structure from 2.1.2

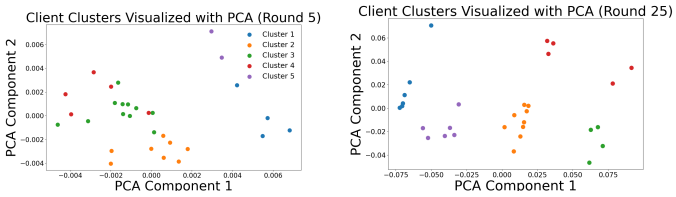
Algorithm 1 MMFL-Group Sampling

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

III. Experiments

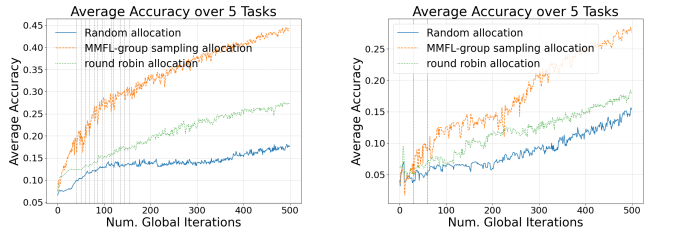
A. Experiment Setup

We conduct 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, result 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. 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 2 convolutional layers, 2 pooling layers, and 2 linear layers, and with different output layer sizes. For the CIFAR-10 and MNIST task, we use a pre-activation ResNet [20]. All experiments are performed for 5 random seeds, and the average is taken.



(a) (b)

Fig. 3: 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) Dynamic Arrival of Clients (b) Dynamic Arrival of Tasks

Fig. 4: Test accuracy for the dynamic cases. Case (a) contains 24 static clients and 36 new clients. Case (b) contains 60 clients in total.

B. Evaluation

We compare our algorithm with two **baseline methods**: *random* allocation and the MMFL round-robin algorithm [21]. Compared to our method, the round-robin algorithm randomly divides clients into 5 groups without any clustering. The random allocation algorithm selects 1/6 of total clients and assigns a model to each client, all in a random manner.

1) Static MMFL system: We first present the results of the consistent cluster structure scenario (Section II-A1) in Figure 2a. 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 [15]'s theoretical analysis. We provide visualizations in Figure 3 to illustrate the cluster structure 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 [15].

We further conduct an experiment with completely independent model cluster structures to challenge our algorithm in the worst case. In this case, the unified cluster structure for all models is very complex or may even not exist, applying original C-Step leads to almost random cluster result. Therefore, we adopt the modified C-Step as discussed in Section II-A2. Our experiment setup is as follows: we still partition each task into 5 ground truth groups based on label clusters. For each task, we randomly select clients and assign them data points corresponding to a specific cluster, and we repeat for each task, therefore creating independent cluster structure for each task. As shown in Figure 2b, the results illustrated a noticeable drop in accuracy across all models compared to the outcomes in Figure 2a. This decline can be attributed to the increased complexity of the cluster structures. However, MMFL-Group Sampling method continues to exhibit strong performance with at least 4% increase of accuracy among all models.

2) Dynamic MMFL System: In Figure 4a, 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 $\lambda = 0.1$ and the client arival times arrival are indicated by the dashed vertical lines in Figure 4a. Compared to Figure 2a, which features only 30 clients, the performance of random and round-robin methods remains similar or worsens despite having additional clients. Conversely, MMFL-Group sampling shows improved performance 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 4b, 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 [22].

IV. Conclusion

In this work, we present the MMFL-Group Sampling algorithm that addresses the data heterogeneity issue in the MMFL system. We incorporate a loss-based grouping mechanism to group clients together and adopt a round-robin way of allocating models to each group. The algorithm is appliable in a practical MMFL system with arrivals of new clients and models. Our algorithm has empirically proved the efficiency of handling cases where group structure is both consistent and inconsistent. We also show an advantage in improving system scalability 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] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[6] A. Hard, K. Partridge, C. Nguyen, N. Subrahmanya, A. Shah, P. Zhu, I. L. Moreno, and R. Mathews, "Training keyword spotting models on non-iid data with federated learning," *arXiv preprint arXiv:2005.10406*, 2020.

[7] D. Guliani, F. Beaufays, and G. Motta, "Training speech recognition models with federated learning: A quality/cost framework," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3080–3084.

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

[9] 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, pp. 906–916.

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

[11] 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.

[12] 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.

[13] 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

[14] 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, pp. 2006–2015.

[15] 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, pp. 5677–5721.

[16] 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.

[17] 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.

[18] 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.

[19] 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, 1967, pp. 281–297.

[20] 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.

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

[22] "Technical report." [Online]. Available: https://tinyurl.com/mmfl-group