# Towards Optimal Heterogeneous Client Sampling in Multi-Model Federated Learning

Haoran Zhang*, Zejun Gong*, Zekai Li*, 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
{haoranz5, zejung, zekail, cjoewong}@andrew.cmu.edu, marie_siew@sutd.edu.sg, rachid.elazouzi@univ-avignon.fr

***Abstract*—Federated learning (FL) enables multiple edge devices to collaboratively train a model without sharing local data. With FL's growing popularity, clients may train multiple unrelated FL models concurrently, known as multi-model federated learning (MMFL). Most FL research does not consider concurrent training of multiple models, and those that do typically use simplified allocations of clients to models with rigid client resource constraints. This paper considers an MMFL system with more flexible resource constraints, where clients have heterogeneous computational abilities and available data to train different models. We start by designing a learning process for a heterogeneous MMFL system, proving it guarantees unbiased convergence. Motivated by this analysis, we propose two client-model allocation methods to accelerate convergence for all models. Our analysis shows these methods perform well, but their client participation variability slows convergence. To mitigate this, we propose a variant of our aggregation method that reuses stale client updates. Finally, we propose a client sampling variant ensuring fair training progress for models with heterogeneous difficulty levels. Extensive experiments show our MMFL algorithms outperform several baselines. Our methods achieve up to 23.4% higher average accuracy compared to random allocation, with only a 4% gap from the theoretically best performance (full participation).**

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

## I. Introduction

Federated Learning (FL) [1] is an increasingly popular distributed learning paradigm that enables clients to collaboratively train a deep learning model under a central server's coordination, with clients' private data remaining locally at the client side. Specifically, in an FL system, a central server maintains a global model and periodically receives updates of model weights from clients within the system. The server aggregates these updates to improve the global model. In this paper, we primarily consider clients that are edge devices, such as smartphones or IoT (Internet-of-Things) devices.

Most FL research assumes each client trains only one model. However, in practice some clients can concurrently train multiple FL models. For example, a smartphone could concurrently act as a client in training Google keyboard prediction [2], [3], keyword-spotting [4], speech recognition [5], recommendation system [6], etc. In this case, building a system that includes multiple FL models, rather than multiple independent single-model FL systems, can lead to better overall training performance. We call an FL system that trains multiple FL models concurrently a *multi-model FL* (MMFL) system. Naïve approaches to MMFL include random client-model allocation and sequential training, where all active clients train the same model in each round, or each client trains all models in each global round. But such naïve methods increase the total training time linearly with the number of models [7]. In contrast, training all FL models concurrently with a well-designed allocation of clients to models could greatly accelerate the convergence for all models [7]–[15].

**Research Challenges in MMFL Systems.** MMFL systems may include numerous FL models. Due to the resource constraints of edge computing devices, some clients may only be able to concurrently train a limited number of FL models in each single round of training. Existing work in MMFL [7]–[15] considers this local computation limitation by assuming that each client can only train *one model* per round and then attempts to improve the average or minimum accuracy of the whole system by effectively assigning tasks to clients. Although this assumption accounts for computation limitations, it fails to capture the complexity of edge devices within the system. Resources (storage, RAM, CPU/GPU, network speed, etc.) often vary across clients, e.g., between smartphones of different ages, resulting in variable and heterogeneous computational capacities between clients within an MMFL system. To the best of our knowledge, *we are the first to consider that clients can have different computational capacities in MMFL*, leading to different clients being able to train different numbers of models concurrently in a single training round. Additionally, some clients might not have the datasets for all models within the system, leading to their differing availability towards model training. Without careful design of the learning and client-model allocation, such heterogeneity can lead to a convergence bias towards more "powerful" clients with high compuational ability and higher availability.

To make matters more challenging, communication constraints on the server side may limit the number of local updates the server can receive from clients in each training round, including the training models and other feedback required for client-model allocation [16]. This restriction means only a limited number of clients can update each model per training round. Such partial client participation introduces additional challenges to the convergence of the training process. Moreover, as the number of models in MMFL increases, an

even smaller fraction of the total clients can be assigned to each model in every round. This results in increased variance in client participation across all models, leading to higher variance that slows convergence [17] and further highlighting the need for an effective strategy for allocating clients to models. We also refer to this allocation as *client sampling*, as it generalizes the well-studied process of sampling clients to train a model in each round in single-model FL [18], [19].

The presence of multiple models, especially if they require different amounts of resources to achieve similar performance, also raises the challenge of fairness in MMFL. Previous works [10], [11] have addressed this fairness problem by proposing sampling algorithms that allocate more clients to poorly performing models. However, they do not consider heterogeneity in *client* capabilities, which introduces considerable complexity into the client-model allocation problem.

**Our Contributions.** Given the challenges of building an effective MMFL system with client heterogeneity in computing, data, communication, and training models, we outline related work in Section II and then make the following contributions:

- We first assume a given client-model assignment and **analyze the resulting convergence of MMFL**, deriving a convergence upper bound with three terms critical for accelerating MMFL convergence. Our bound reveals the significant effect of client-model assignment in MMFL. As the assignment strategy in MMFL may introduce training bias, we provide a method that provably mitigates this bias, ensuring unbiased convergence (Section III).
- Based on our convergence analysis, we propose **gradient-based and loss-based optimal variance-reduced sampling algorithms** (**MMFL-GVR** and **MMFL-LVR**). These algorithms use different metrics to identify "important" clients in each round and prioritize allocating such clients to models, minimizing the variance of global model updates under server communication and heterogeneous client computation constraints (Section IV).
- MMFL-GVR and MMFL-LVR can generate high participation heterogeneity between clients for each model, leading to increased variance and slower convergence. To reduce this variance, we propose **MMFL-GVR***, which reuses stale client updates in the model aggregation, while maintaining unbiased training (Section V).
- We introduce a novel method (**MMFL-FairVR**) that promotes fair model training while accounting for the heterogeneity of models and clients (Section VI).
- We conduct **extensive experiments** in various settings on real-world datasets, demonstrating that our methods significantly outperform other baselines. By adopting aggregation with stale updates to mitigate the impact of participation heterogeneity, MMFL-GVR* achieves 94% of the theoretical best performance under the same experiment settings. Additionally, MMFL-FairVR achieves 33.8% higher minimum accuracy across all models compared to the *random* allocation (Section VII).

We conclude the paper in Section VIII.

## II. Related Work

We give an overview of two lines of FL research that are most relevant to our work.

**Multi-Model Federated Learning.** Multi-model FL was first proposed in [8]. Many subsequent studies [7]–[15], [20] have sought to improve the training performance of all models by effectively allocating the available training resources (clients) across models. This resource allocation problem can be solved via multi-armed bandits [8], using reinforcement learning [12], [13], or directly constructing the probability distribution of training each model [10], [11]. Other works [7], [14] address variations in model complexity and local training times by proposing asynchronous MMFL training schemes. While these works improve the average or minimum accuracy across MMFL models, they generally overlook heterogeneity in clients' computational capabilities and dataset distributions. Client heterogeneity in MMFL introduces new client-model allocation challenges because: 1) Clients impose different communication costs at the server, as clients that train more models must communicate all of their updates to the server; and 2) The system may converge towards clients with more powerful computational abilities since they can contribute to more models per round, causing convergence bias.

**Single-Model Client Sampling.** Allocating clients to models in MMFL can be viewed as sampling a set of clients to train each model in each round, generalizing the well-studied client sampling in single-model FL [18], [19], [21]–[26]. These strategies generally define metrics that evaluate the "importance" of clients to the training task and prioritize sampling accordingly. Some naïve methods [18], [19] can be easily adapted for MMFL settings, while others [21]–[26] cannot. Naïve methods [18], [19] rank clients based on their importance metrics and select the top ones. However, they may introduce training bias, as some clients are selected more frequently than others, leading to a non-vanishing bias in the final model [18], [27]. To address sampling bias, more sophisticated methods [21]–[25] also adjust the aggregation coefficients according to their constructed client sampling distribution, theoretically avoiding bias. However, these methods cannot be directly applied to MMFL because they generate only the probability of sampling a client; MMFL also requires determining which model a client should train. Client-model allocation distributions in previous MMFL works [10], [11], are uniform across all clients, ignoring client heterogeneity and leaving significant room for improvement.

## III. System Model

In MMFL (see Fig. 1), we assume $S$ models are trained iteratively across global rounds indexed by $\tau = 1, 2, \ldots, T$. Within each global round $\tau$, the server allocates clients to each model $s = 1, 2, \ldots, S$. Clients then train each of their assigned models locally, as in single-model FL. After the local training, clients upload their local model updates to the server, where they are aggregated for each model, and the process repeats.

### A. Problem Formulation

Similar to previous MMFL works [7]–[15], we consider an MMFL system with $N$ clients and $S$ unrelated models. Define $\mathcal{S}$ as the set of models. We model clients' *heterogeneous computational abilities* by assuming that each client $i$ can train $B_i$ models per global training round. For ease of description, we say that client $i$ has $B_i$ **processors** to handle FL training tasks, where a "training task" means performing local training for a model within one global round. Define $\mathcal{B}_i$ as the set of processors of client $i$.[1] This model generalizes that of previous MMFL works [7]–[15], which assume a client can only train one model per round, i.e., $B_i = 1$ for all clients $i$. While models may need different amounts of training resources, for simplification, we suppose that the batch size of each model is adjusted to ensure that each model requires roughly the same amount of computing resources for clients in each global round.[2] Since MMFL includes multiple unrelated FL training tasks within one system, some clients *may lack the datasets for specific models*, which further complicates resource allocation, given that different clients have different amounts of tasks they are involved in. We define the set of clients available for model $s$ as $\mathcal{N}_s$, and the set of models available for client $i$ as $\mathcal{S}_i$. We define the **objective for each model** $s$ as:

$$\min_{w_s} F_s = \min_{w_s} \sum_{i \in \mathcal{N}_s} d_{i,s} f_{i,s}(w_s). \tag{1}$$

where $w_s$ denotes the parameters of model $s$. The local objective $f_{i,s}(w_s)$ can be defined by empirical risks over local data: $f_{i,s}(w_s) = \frac{1}{n_{i,s}} \sum_{\xi \in \mathcal{D}_{i,s}} l(w_s, \xi)$, where $\mathcal{D}_{i,s}$ is the set of datapoints in client $i$ that can be used to train model $s$, and $n_{i,s} = |\mathcal{D}_{i,s}|$ is the number of datapoints that client $i$ can use to train model $s$. The loss function $l$ evaluates model performance on one datapoint, e.g., with cross-entropy loss. $d_{i,s} = n_{i,s} / \sum_{j \in \mathcal{N}_s} n_{j,s}$ denotes the fraction of client $i$'s dataset size relative to the total dataset size for model $s$. Based on (1), the objective of the MMFL system is to minimize the sum of the objectives of the $S$ models, i.e.,

$$\min_{w_1, \cdots, w_S} F = \min_{w_1, \cdots, w_S} \sum_{s=1}^{S} \sum_{i \in \mathcal{N}_s} d_{i,s} f_{i,s}(w_s) \tag{2}$$

### B. MMFL Training Procedure

Before the local training in each round, the server will firstly assign training tasks to clients. The task allocation process consists of assigning $c$ ($c \leq B_i$) training tasks to each client $i$ in each round. We define $p^{\tau}_{s|(i,b)}$ as the probability of assigning processor $b$ at client $i$, which we denote as processor $(i,b)$, to train model $s$ in global round $\tau$. Thus, the server samples from the distribution $\mathbf{p}^{\tau} = \{p^{\tau}_{s|(i,b)}\}_{s \in \mathcal{S}, i \in \mathcal{N}_s, b \in \mathcal{B}_i}$ in assigning tasks to client processors. This probability distribution is designed by the server (Section IV) and will play an important role in accelerating convergence and implementing fairness.

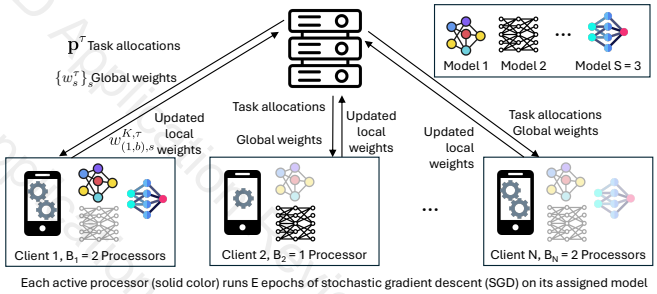


Fig. 1. Overview of the MMFL system for an example with $S = 3$ models. In each global round $\tau$, the server probabilistically assigns models to a subset of processors at the FL clients. Models that each client has the data to train are shown at each client, and faded models indicate ones that have not been assigned in this training round.

After the training task allocation, active clients execute the assigned tasks and send updates to the server for aggregation. The detailed steps in each global round $\tau$ are described below:

- **Training Task Allocation.** Given the probability distribution $\mathbf{p}^{\tau}$, the server generates the sets of participating processors for each model $s$, $(\mathcal{A}_{\tau,s})_{s \in \mathcal{S}}$ as follows: for each $s \in \mathcal{S}$ and $i \in \mathcal{N}_s$, we include $(i,b) \in \mathcal{A}_{\tau,s}$ with probability $p^{\tau}_{s|(i,b)}$. The assignment of each processor to a task $s$ is independent of other processors.
- **Synchronization.** Each processor $(i,b) \in \mathcal{A}_{\tau,s}$, $s \in \mathcal{S}$, initializes the model weights with the global model $s$ weights $w_s^{\tau}$: $w^{1,\tau}_{(i,b),s} = w_s^{\tau}$, where $w^{1,\tau}_{(i,b),s}$ denotes processor $(i,b)$'s initial weights of model $s$ in round $\tau$.
- **Local Training.** Each processor $(i,b) \in \mathcal{A}_{\tau,s}$, $s \in \mathcal{S}$, performs local training by running mini-batch stochastic gradient descent for $K$ local epochs: $w^{t+1,\tau}_{(i,b),s} = w^{t,\tau}_{(i,b),s} - \eta_{\tau} \nabla f_{i,s}(w^{t,\tau}_{(i,b),s}, \xi^{t,\tau}_{i,s})$ for $t = 1, 2, \ldots, K$, where $\xi^{t,\tau}_{i,s}$ represents a random mini-batch of datapoints from local data distribution $\mathcal{D}_{i,s}$ and $\eta_{\tau}$ the local learning rate. Define the change in model weights produced by processor $(i,b)$ as $G^{\tau}_{(i,b),s} = \eta_{\tau} \sum_{t=1}^{K} \nabla f_{i,s}(w^{t,\tau}_{(i,b),s}, \xi^{t,\tau}_{i,s})$.
- **Aggregation.** Clients send their local updates to the server, which aggregates the weights for each model $s$:

$$w_s^{\tau+1} = w_s^{\tau} - \sum_{(i,b) \in \mathcal{A}_{\tau,s}} P^{\tau}_{(i,b),s} G^{\tau}_{(i,b),s}, \tag{3}$$

where $P^{\tau}_{(i,b),s} = \frac{d_{i,s}}{B_i p^{\tau}_{s|(i,b)}}$. The aggregation coefficient $P^{\tau}_{(i,b),s}$ acts as a scaling factor to guarantee an unbiased estimator for full participation training, i.e.,

[1] In real applications, each training task need not be assigned to a dedicated client processor; we use the concept of a *processor* as a useful abstraction of clients' heterogeneous computational abilities.

[2] Alternatively, it is not difficult to extend our approach to sort models into "types" that quantify the resources needed to train each model.

$$\mathbb{E}\left[\sum_{(i,b)\in\mathcal{A}_{\tau,s}} P^{\tau}_{(i,b),s} G^{\tau}_{(i,b),s} \,\middle|\, G^{\tau}_{(i,b),s}\right] \tag{4}$$

$$= \sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i} \mathbb{E}\left[\mathbb{1}_{(i,b)\in\mathcal{A}_{\tau,s}}\right] \frac{d_{i,s}}{B_i p^{\tau}_{s|(i,b)}} G^{\tau}_{(i,b),s} \tag{5}$$

$$= \sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i} \frac{d_{i,s}}{B_i} G^{\tau}_{(i,b),s} \tag{6}$$

Here $\mathbb{1}_{(i,b)\in\mathcal{A}_{\tau,s}}$ is an indicator of whether processor $(i,b)$ is allocated to model $s$ for training in round $\tau$. We simplify this notation to $\mathbb{1}^{s,\tau}_{(i,b)}$ in the following. The above expectation is taken over $\mathcal{A}_{\tau,s}$ (set of participating processors). Eq. 6 is full participation update using mini-batch stochastic gradient descent (SGD). *FedAvg* [1] can be viewed as a special case of our MMFL system with $S=1$ and $B_i=1$ for all clients.

**Remark 1** (Independent processor sampling)**.** *Independent client sampling is widely adopted in single-model sampling methods [21]–[25]. In MMFL, with heterogeneous computational abilities, independent sampling at the* ***processor level*** *helps manage participating clients given resource constraints and theoretically avoids training bias. Since we assume each processor's sampling is independent, $l$ processors ($1 \le l \le B_i$) in client $i$ could train the same model. In practice, client $i$ can train with one processor, and upload $l \cdot G^{\tau}_{(i,b),s}$ as its update to the server.*

Before designing optimal sampling methods with optimized distribution $\mathbf{p}^{\tau}$, we study MMFL convergence and highlight dominant terms critical for accelerating convergence. These terms will be used as objective functions in our optimization problems to design optimal $\mathbf{p}^{\tau}$ for client-model allocation.

### C. Convergence Analysis

We analyze the convergence of the MMFL system described above, revealing that $\mathbf{p}^{\tau}$ can significantly influence the convergence speed. This analysis involves several assumptions. Assumptions 2-5 are standard in FL literature [18]. Assumption 6 ensures a lower-bounded probability to prevent extreme $\mathbf{p}^{\tau}$ from causing some clients to disappear entirely from the training. In Definition 1, we quantify clients' non-iid data distribution, which is common in real-world FL applications.

**Assumption 2** ($L$-smoothness)**.** *Each $f_{i,s}$ is L-smooth, and thus $F=\sum_{s=1}^{S}\sum_{i\in\mathcal{N}_s} d_{i,s} f_{i,s}$ is also L-smooth.*

**Assumption 3** (Strong convexity)**.** *Each $f_{i,s}$ is $\mu$-strongly convex, and thus $F=\sum_{s=1}^{S}\sum_{i\in\mathcal{N}_s} d_{i,s} f_{i,s}$ is as well.*

**Assumption 4** (Bounded variance)**.** *The variance of the mini-batch gradients is bounded: $\mathbb{E}_{\xi_{i,s}\sim\mathcal{D}_{i,s}}(\|\nabla f_{i,s}(w,\xi_{i,s}) - \nabla f_{i,s}(w)\|^2) \le \sigma^2_{i,s},\ \forall i,s.$*

**Assumption 5** (Bounded gradients)**.** *The expected squared norm of the gradients is uniformly bounded: $\mathbb{E}_{\xi_{i,s}\sim\mathcal{D}_{i,s}}(\|\nabla f_{i,s}(w,\xi_{i,s})\|^2) < \bar{\sigma}^2,\ \forall i,s.$*

**Assumption 6** (Lower-bounded probability)**.** *There exists a lower bound $\theta>0$, such that $p^{\tau}_{s|(i,b)} \ge \theta$ for all $i,b,s,\tau$.*

**Definition 1** (Non-iid data distribution)**.** *We quantify the non-iid level of clients' data distribution as: $\Gamma_{i,s} = f_{i,s}(w^*_s) - f_{i,s}(w^{*,i}_s)$, where $w^*_s$ minimizes the objective $F_s$ and $w^{*,i}_s$ minimizes $f_{i,s}$ for client $i$. There exists a gap between the global optimal and the local optimal weights: $\|w^*_s - w^{*,i}_s\| \ge e_w > 0$, which also implies $\|\nabla f_{i,s}(w^{\tau}_s)\| \ge e_f > 0$ for all $i,s,\tau$.*

**Lemma 7** (Local loss and client participation bound)**.** *We provide an upper bound for a loss-related term:*

$$\frac{\left(\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\mathbb{1}^{s,\tau}_{(i,b)}P^{\tau}_{(i,b),s}-1\right)^2}{\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{1}{(f_{i,s}(w^{\tau}_s))^2}} \tag{7}$$

$$\le \sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\left(\mathbb{1}^{s,\tau}_{(i,b)}P^{\tau}_{(i,b),s} - \frac{d_{i,s}}{B_i}\right)^2 (f_{i,s}(w^{\tau}_s))^2 \tag{8}$$

*Proof Sketch*: Replace 1 in (7) using $\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{d_{i,s}}{B_i}$ and apply *Titu's lemma* [28] to upper bound the resulting term. □

**Theorem 8** (Convergence)**.** *Let $w^*_s$ denote the optimal weights of model $s$. If the learning rate $\eta_{\tau}=\frac{16}{\mu}\frac{1}{(\tau+1)K+\gamma}$, then*

$$\mathbb{E}\left(\|w^{\tau}_s - w^*_s\|^2\right) \le \frac{V_{\tau}}{(\tau K+\gamma_{\tau})^2} \tag{9}$$

*Here we define* $\gamma_{\tau} = \max\{\frac{32L}{\mu}, 4K\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\mathbb{1}^{s,\tau}_{(i,b)}P^{\tau}_{(i,b),s}\}$
$V_{\tau} = \max\{\gamma^2\mathbb{E}(\|w^0_s - w^*_s\|^2), (\frac{16}{\mu})^2\sum_{\tau'=0}^{\tau-1} z_{\tau'}\}$,
$z_{\tau'} = \mathbb{E}[Z_g^{\tau'} + Z_l^{\tau'} + Z_p^{\tau'}]$,
$\mathbb{E}[Z_g^{\tau}] = K\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{(\frac{d_{i,s}}{B_i}\sigma_{i,s})^2}{p^{\tau}_{s|(i,b)}} + 4LK\sum_{i\in\mathcal{N}_s} d_{i,s}\Gamma_{i,s} + \max(\frac{B_i}{d_{i,s}})\mathbb{E}[\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{(\frac{d_{i,s}}{B_i})^2\sum_{t=1}^{K}\|\nabla f_{i,s}(w^{t,\tau}_{(i,b),s})\|^2}{p^{\tau}_{s|(i,b)}}]$,
$\mathbb{E}[Z_l^{\tau}] = R\mathbb{E}[V_s\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}(\mathbb{1}^{s,\tau}_{(i,b)}P^{\tau}_{(i,b),s}f_{i,s}(w^{\tau}_s) - \frac{d_{i,s}}{B_i}f_{i,s}(w^{\tau}_s))^2]$, *where* $R=\frac{2K^3\bar{\sigma}^2}{e_w^2 e_f^2\theta}, V_s=\sum_{i\in\mathcal{N}_s}B_i$,
$\mathbb{E}[Z_p^{\tau}] = (\frac{2}{\theta} + K(2 + \frac{\mu}{2L}))K^2\bar{\sigma}^2 + \frac{2K^3\bar{\sigma}^2}{\theta}\mathbb{E}[(\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\mathbb{1}^{s,\tau}_{(i,b)}P^{\tau}_{(i,b),s}-1)^2]$.

*Proof Sketch*: We modified and adapted the proof from [29], which also considered client heterogeneity. Unlike our model of assuming client $i$ can handle $B_i$ training tasks in each global round, however, [29] modeled different computational abilities by varying the number of local epochs that clients could afford, and in a single-model FL system.

In [29], aggregation coefficients are adjusted based on clients' computational abilities. Similarly, we adjust them using each client's computational ability ($B_i$) and sampling distribution ($\mathbf{p}^{\tau}$) to avoid bias. We first extend their proof from single-model FL to MMFL and modify it to account for varying $B_i$ and $\mathbf{p}^{\tau}$. While [29]'s original theorem shows unbiased training, it does not analyze the impact of different local epochs on convergence speed. In our setting, $B_i$ is fixed by the client, while $\mathbf{p}^{\tau}$ can change in each round. To analyze the impact of $\mathbf{p}^{\tau}$ on convergence speed, we make the following modifications: 1) Instead of bounding $\|\nabla f_{i,s}(w^{t,\tau}_{(i,b),s})\|$ with

$\bar{\sigma}$, we keep $\|\nabla f_{i,s}(w^{t,\tau}_{(i,b),s})\|$ in some specific steps. 2) Instead of bounding a loss-related term with a constant, we introduce Lemma 7 to obtain a tighter, more informative bound. □

*Interpretation:* Considering $B_i$ processors for each client is analogous to having $B_i$ clients with identical datasets, with each client's contribution reduced by $\frac{1}{B_i}$. Equation 9's bound is mainly influenced by the terms $\mathbb{E}[Z_g^\tau]$, $\mathbb{E}[Z_l^\tau]$, and $\mathbb{E}[Z_p^\tau]$.

In $\mathbb{E}[Z_g^\tau]$, the first two terms reflect the influence of the non-iid data and the variance of the mini-batch gradient ($\sigma_{i,s}$) on the convergence upper bound. These terms are determined by datasets and local training batch size. If the data is very non-iid across clients, or the mini-batch gradient is inaccurate, $\mathbb{E}[Z_g^\tau]$'s value increases, slowing convergence speed. The third term, $\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{(\frac{d_{i,s}}{B_i})^2\sum_{t=1}^{K}\|\nabla f_{i,s}(w^{t,\tau}_{(i,b),s})\|^2}{p^\tau_{s|(i,b)}}$, indicates that the relationship between the norm of local updates and sampling probability distribution could influence the convergence speed. We notice that this term reflects the scale of the **variance of sampled updates** during training. First, we define the variance of sampled updates as:

$$\sum_{s=1}^{S}\mathbb{E}\left[\|\sum_{(i,b)\in\mathcal{A}_{\tau,s}}P^\tau_{(i,b),s}\frac{G^\tau_{(i,b),s}}{\eta_\tau}-\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{d_{i,s}}{B_i}\frac{G^\tau_{(i,b),s}}{\eta_\tau}\|^2\right] \tag{10}$$

Since the expectation of the sampled update (left term) is the full participation update,[3] Eq. (10) represents the variance of the sampled update. Using the fact that sampling at each processor is independent of other processors, we now show that the term $\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{(\frac{d_{i,s}}{B_i})^2\sum_{t=1}^{K}\|\nabla f_{i,s}(w^{t,\tau}_{(i,b),s})\|^2}{p^\tau_{s|(i,b)}}$ reflects the scale of the variance of sampled updates:

$$\mathbb{E}[\|\sum_{(i,b)\in\mathcal{A}_{\tau,s}}P^\tau_{(i,b),s}\frac{G^\tau_{(i,b),s}}{\eta_\tau}-\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{d_{i,s}}{B_i}\frac{G^\tau_{(i,b),s}}{\eta_\tau}\|^2] \tag{11}$$

$$=\sum_{(i,b),(j,v)}\frac{d_{i,s}d_{j,s}(G^\tau_{(i,b),s})^\top G^\tau_{(j,v),s}}{\eta_\tau^2B_iB_jp^\tau_{s|(i,b)}p^\tau_{s|(j,v)}}\mathbb{E}[\mathbb{1}_{(i,b),(j,v)\in\mathcal{A}_{\tau,s}}]$$
$$-\sum_{(i,b),(j,v)}\frac{d_{i,s}d_{j,s}}{\eta_\tau^2B_iB_j}(G^\tau_{(i,b),s})^\top G^\tau_{(j,v),s}$$

$$=\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{\|\frac{d_{i,s}}{B_i\eta_\tau}G^\tau_{(i,b),s}\|^2}{p^\tau_{s|(i,b)}}-\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\|\frac{d_{i,s}G^\tau_{(i,b),s}}{B_i\eta_\tau}\|^2 \tag{12}$$

The distribution $\mathbf{p}^\tau$ does not change the scale of the second term, and $K\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{(\frac{d_{i,s}}{B_i})^2\sum_{t=1}^{K}\|\nabla f_{i,s}(w^{t,\tau}_{(i,b),s})\|^2}{p^\tau_{s|(i,b)}}$ is the upper bound for the first term. Thus, this term implies that the variance of sampled updates could impact the convergence speed. In the next section, we choose to directly minimize the variance of sampled updates for all models instead of directly minimizing this term.

$\mathbb{E}[Z_l^\tau]$ shows that the relationship between local loss values $f_{i,s}(w_s^\tau)$ and aggregation coefficients $P^\tau_{(i,b),s}$ also influences convergence speed. Using the triangle inequality, this term can be viewed as the upper bound of $\mathbb{E}[(\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\mathbb{1}^{s,\tau}_{(i,b)}P^\tau_{(i,b),s}f_{i,s}(w_s^\tau)-\sum_{i\in\mathcal{N}_s}d_{i,s}f_{i,s}(w_s^\tau))^2]$, representing the variance of the surrogate objective $\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\mathbb{1}^{s,\tau}_{(i,b)}P^\tau_{(i,b),s}f_{i,s}(w_s^\tau)$ that clients implicitly optimize for model $s$ in each round.

In $\mathbb{E}[Z_p^\tau]$, the first term can be viewed as a constant. The second term reveals that the variance of aggregation coefficients also affects the convergence speed. If sampling probabilities are too heterogeneous across clients, this could lead to unstable training despite equal client contributions.

## IV. Optimal Variance-Reduced Sampling

In this section, motivated by reducing the convergence upper bound we derived in Theorem 8, we propose several strategies to generate distribution $\mathbf{p}^\tau$ for training task allocation.

### A. Gradient-Based Optimal Variance-Reduced Sampling

In $\mathbb{E}[Z_g^\tau]$, we reveal the impact of the variance of the sampled update on the convergence speed. Thus, we optimize $\mathbf{p}^\tau$ to minimize the variance of the sampled update by solving

$$\min_{\{p^\tau_{s|(i,b)}\}}\sum_{s=1}^{S}\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{\|\frac{d_{i,s}}{B_i\eta_\tau}G^\tau_{(i,b),s}\|^2}{p^\tau_{s|(i,b)}} \tag{13}$$
$$\text{s.t. } p^\tau_{s|(i,b)}\geq 0,\ \sum_{s\in\mathcal{S}_i}p^\tau_{s|(i,b)}\leq 1,\ \sum_{s=1}^{S}\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}p^\tau_{s|(i,b)}=m,$$
$$\forall i,b,s,\tau.$$

The first two constraints in this optimization problem ensure a feasible probability distribution for each processor $(i,b)$. The third constraint ensures that the server expects to receive $m$ total local updates from all clients, by ensuring that $m$ training tasks are assigned to the clients on expectation. This constraint thus models the server's communication limitations, e.g., if the server is at an edge base station, base stations have an upper bound on the number of connections[4] they can maintain. In general, allowing more connections (a higher value of $m$) also implies a need for more sophisticated queuing and parallel processing capabilities at the server, especially if there are thousands of clients and processors, as may be the case in edge FL systems [1]. Intuitively, a high value of $m$ will lead to faster convergence but also higher costs.

**Theorem 9** (Optimal assignment probabilities)**.** *Equation* (13)*'s optimization problem is solved by setting:*

$$p^\tau_{s|(i,b)}=\begin{cases}\frac{(m-V+k)\|\tilde{U}^\tau_{(i,b),s}\|}{\sum_{(j,v)\notin\mathcal{V}_{s,0}}M^\tau_{(j,v)}} & (i,b)\notin\mathcal{V}_{s,0},\\ \frac{\|\tilde{U}^\tau_{(i,b),s}\|}{M^\tau_{(i,b)}} & (i,b)\in\mathcal{V}_{s,0}.\end{cases} \tag{14}$$

[3]Notice $\mathbb{E}\left[\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\frac{d_{i,s}}{B_ip^\tau_{s|(i,b)}}\frac{G^\tau_{(i,b),s}}{\eta_\tau}\right]=\sum_{i\in\mathcal{N}_s}d_{i,s}\frac{G^\tau_{i,s}}{\eta_\tau}$

[4]We can easily extend our formulation to server bandwidth constraints by constraining the expected size of the models sent to the server.

**Algorithm 1** MMFL-GVR

**Input:** expected number of training tasks $m$, the set of available clients for each model: $\mathcal{N}_s$, computation ability $B_i$ for each client
each client $i$ computes update $G^\tau_{(i,b),s}$ for all models (in parallel)
each client $i$ sends $\|\frac{d_{i,s}}{B_i\eta_\tau}G^\tau_{(i,b),s}\|$ to the server
server generates $\mathbf{p}^\tau$ using Theorem 9
server generates model allocation from $\mathbf{p}^\tau$ and requests updates
active client $i$ sends its updates to server
server conducts aggregation using Eq. 3

*where* $V = \sum_{i\in\mathcal{N}_1\cup\cdots\mathcal{N}_S} B_i$, $\|\tilde{U}^\tau_{(i,b),s}\| = \|\frac{d_{i,s}}{B_i\eta_\tau}G^\tau_{(i,b),s}\|$, $M^\tau_{(i,b)} = \sum_{s\in\mathcal{S}_i}\|\tilde{U}^\tau_{(i,b),s}\|$. $\mathcal{V}_{s,0}$ *is the smallest set satisfying*

$$0 < (m - V + k) \leq \frac{\sum_{(j,v)\notin\mathcal{V}_{s,0}} M^\tau_{(j,v)}}{\min_{(i,b)\in\mathcal{V}_{s,0}}[M^\tau_{(i,b)}]} \tag{15}$$

*and* $k = |\mathcal{V}_{s,0}|$. *The set* $\mathcal{V}_{s,0}$ *includes the top* $k$ *processors with the largest* $M^\tau_{(i,b)}$.

*Proof Sketch:* We solve the problem (Eq. (13)) by designing the Lagrangian function and using the KKT conditions. □

*Interpretation:* By minimizing the variance in Eq. (13), the direction of the update with sampled processors ($(i,b)\in\mathcal{A}_{\tau,s}$) should be more stable and closer to the update with full participation, i.e., if all processors compute an update for model $s$, which eliminates the convergence slowdown due to partial client participation. From Theorem 9, the closed-form solution of $p^\tau_{s|(i,b)}$ shows that the server assigns higher probability to clients whose updates have larger $\ell_2$ norms. Intuitively, such client updates will dominate the update with full client participation, as a larger model update indicates that the model likely performs poorly on this client's data.

We name the **G**radient-based optimal **V**ariance-**R**educed Sampling Algorithm that uses Eq. (13)'s allocation solution as **MMFL-GVR**, whose pseudocode is shown in Algorithm 1. Although MMFL-GVR reduces the gap between the sampled and full participation update, it requires all clients to train all models in each round to obtain the gradient norms for each model, which can impose a large computation burden.

### *B. Loss-Based Optimal Variance-Reduced Sampling*

To avoid the burden of having all clients train all models in each round, we next introduce **L**oss-based optimal **V**ariance-**R**educed Sampling (**MMFL-LVR**). Theoretically, **MMFL-LVR** optimizes the term $\mathbb{E}[Z_l^\tau]$ in Theorem 8 at time $\tau$, which is the variance of the surrogate objective ($\mathbb{E}[(\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}\mathbb{1}^{s,\tau}_{(i,b)}P^\tau_{(i,b),s}f_{i,s}(w^\tau_s) - \sum_{i\in\mathcal{N}_s}d_{i,s}f_{i,s}(w^\tau_s))^2]$) that the sampled clients implicitly optimize. Formally, we aim to solve:

$$\min_{\mathbf{p}^\tau}\sum_{s=1}^{S}\mathbb{E}[(\sum_{(i,b)\in\mathcal{A}_{\tau,s}}P^\tau_{(i,b),s}f_{i,s}(w^\tau_s) - \sum_{i\in\mathcal{N}_s}d_{i,s}f_{i,s}(w^\tau_s))^2] \tag{16}$$

$$\text{s.t. } p^\tau_{s|(i,b)}\geq 0,\ \sum_{s=1}^{S}p^\tau_{s|(i,b)}\leq 1,\ \sum_{s=1}^{S}\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i}p^\tau_{s|(i,b)} = m,$$
$$\forall i,b,s,\tau.$$

The closed-form solution of this problem can be deduced following similar steps as Theorem 9.

**Theorem 10** (Optimal MMFL-LVR assignment probabilities)**.** *Equation (16)'s optimization problem is solved by*

$$p^\tau_{s|(i,b)} = \begin{cases} \frac{(m-V+k)\|\tilde{U}^\tau_{(i,b),s}\|}{\sum_{(j,v)\notin\mathcal{V}_{s,0}}M^\tau_{(j,v)}} & \text{if } (i,b)\notin\mathcal{V}_{s,0},\\ \frac{\|\tilde{U}^\tau_{(i,b),s}\|}{M^\tau_{(i,b)}} & \text{if } (i,b)\in\mathcal{V}_{s,0}.\end{cases} \tag{17}$$

*where* $V = \sum_{i\in\mathcal{N}_1\cup\cdots\mathcal{N}_S}B_i$, $\|\tilde{U}^\tau_{(i,b),s}\| = \|\frac{d_{i,s}}{B_i}f_{i,s}(w^\tau_s)\|$, $M^\tau_{(i,b)} = \sum_{s\in\mathcal{S}_i}\|\tilde{U}^\tau_{(i,b),s}\|$, *and* $m$ *is the number of active processors on expectation.* $\mathcal{V}_{s,0}$ *is the smallest set satisfying*

$$0 < (m - V + k) \leq \frac{\sum_{(j,v)\notin\mathcal{V}_{s,0}} M^\tau_{(j,v)}}{\min_{(i,b)\in\mathcal{V}_{s,0}}[M^\tau_{(i,b)}]} \tag{18}$$

*and* $k = |\mathcal{V}_{s,0}|$. *The set* $\mathcal{V}_{s,0}$ *includes the top* $k$ *processors with the largest* $M^\tau_{(i,b)}$.

**Algorithm 2** MMFL-LVR

**Input:** expected number of training tasks $m$, the set of available clients for each model: $\mathcal{N}_s$, computation ability $B_i$ for each client
each client $i$ computes loss $f_{i,s}(w^\tau_s)$ for all models (in parallel)
each client $i$ sends $\frac{d_{i,s}}{B_i}f_{i,s}(w^\tau_s)$ to the server
server generates $\mathbf{p}^\tau$ using Theorem 10
server generates model allocation from $\mathbf{p}^\tau$, requests updates
active clients conduct local training, send $G^\tau_{(i,b),s}$ to the server
server conducts aggregation using Eq. 3

The pseudocode of MMFL-LVR is shown in Algorithm 2. MMFL-LVR *only requires all clients to upload their local loss values* for the server to compute the sampling distribution $\mathbf{p}^\tau$, which can be accomplished through a forward pass of the current model and requires much less computational resources than the gradient computations needed for MMFL-GVR.

*Lower bound of the probability:* Theorem 8 requires that the probability has a lower bound: $p^\tau_{s|(i,b)} > \theta$. Otherwise, the proof may not ensure convergence as some clients may rarely or never participate in the training. To avoid such cases for our MMFL-LVR and MMFL-GVR, a small constant can be added to the gradient norm $\|\frac{d_{i,s}}{B_i\eta_\tau}G^\tau_{(i,b),s}\|$ or local loss $\frac{d_{i,s}}{B_i}f_{i,s}(w^\tau_s)$, which does not affect the practical distribution but theoretically ensures convergence.

## V. Heterogeneity of Client Participation

We can intuitively understand MMFL-GVR and MMFL-LVR as evaluating the importance of each client based on

specific metrics (gradient norms and local loss values, respectively) and assigning higher assignment probabilities to "more important" clients. However, this approach can increase the heterogeneity of client participation if clients have variable gradient norms or local loss values, e.g., the model performs much better on one client compared to another. Previous work [17] has found that high variance in client participation can slow down convergence, but the underlying reasons for this effect were not clearly explained. Our convergence upper bound highlights this variance in the $\mathbb{E}[Z_p^\tau]$ term ($\mathbb{E}[(\sum_{(i,b)\in\mathcal{A}_{\tau,s}} P^\tau_{(i,b),s} - 1)^2]$).

**Variance in Client Participation.** To see why variance in client participation can slow down convergence, we rewrite the aggregation rule defined in Eq. (3) as:

$$w_s^{\tau+1} = w_s^\tau - H_{\tau,s}^\top G_s \tag{19}$$

where $H_{\tau,s} = [\cdots, \mathbb{1}^{s,\tau}_{(i,b)} P^\tau_{(i,b),s}, \cdots]^\top$, $G_s = [\cdots, G^\tau_{(i,b),s}, \cdots]^\top$. We can compute the expectation of $\|H_{\tau,s}\|_1$ over the sampled processors ($\|\cdot\|_1$ indicates the $\ell_1$ norm):

$$\mathbb{E}[\|H_{\tau,s}\|_1] = \mathbb{E}\left[\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i} \frac{d_{i,s}}{B_i p^\tau_{s|(i,b)}} \mathbb{1}^{s,\tau}_{(i,b)}\right] \tag{20}$$

$$= \sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i} \frac{d_{i,s}}{B_i} = 1 \tag{21}$$

In full participation, where each processor trains all models per round, $\|H_{\tau,s}\|_1 = 1$. And in partial participation, $\mathbb{E}[\|H_{\tau,s}\|_1] = 1$. Thus, $\mathbb{E}[(\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i} P^\tau_{(i,b),s} - 1)^2]$ can be interpreted as the variance of $\|H_{\tau,s}\|_1$:

$$\mathbb{E}\left[(\sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i} P^\tau_{(i,b),s} - 1)^2\right] = \mathbb{E}[(\|H_{\tau,s}\|_1 - \mathbb{E}\left[\|H_{\tau,s}\|_1\right])^2]$$

As shown in Eq. (19), $\|H_{\tau,s}\|_1$ represents the "global step size." A higher variance in $\|H_{\tau,s}\|_1$ indicates instability in this "global step size." Thus, even if the update *direction* under sampling distribution $\mathbf{p}^\tau$ is close to that of full participation, variance in client and processor participation can lead to variance in the *step size* compared to that of full participation, which may affect the stability of the training process. A lower variance stabilizes the sampled update's global *step size* and makes it similar to the *step size* with full participation, ensuring that this step size is not too large or too small.

**Stable Updates with Participation Heterogeneity.** We can mitigate the impact of the high variance of $\|H_{\tau,s}\|_1$ by changing our aggregation rule to the following:

$$w_s^{\tau+1} = w_s^\tau - \Delta_{\tau,s} \tag{22}$$
$$\Delta_{\tau,s} = \sum_{i\in\mathcal{N}_s} d_{i,s} h^\tau_{i,s} + \sum_{(i,b)\in\mathcal{A}_{\tau,s}} \frac{d_{i,s}(G^\tau_{(i,b),s} - h^\tau_{i,s})}{B_i p^\tau_{s|(i,b)}}$$

where $h^\tau_{i,s}$ denotes the last received update as of round $\tau$ from each client $i$ for each model $s$.[5] If client $i$ is active for model $s$ in this round, then $h^\tau_{i,s} = G^{\tau-1}_{i,s} = \mathbb{E}[G^{\tau-1}_{(i,b),s}]$. The expectation of $\Delta_{\tau,s}$ over the processor assignment distribution is equal to a full participation update, ensuring that the training remains unbiased. This aggregation rule was first proposed in [30] to improve single-model FL's performance with a fixed sampling distribution. In our case, the sampling distribution is determined by MMFL-GVR or MMFL-LVR in each round. If the heterogeneity of participation (variance of $\|H_{\tau,s}\|_1$) greatly impacts training stability, the server can adopt this new aggregation rule (Eq. (22)) to ensure a stable global step size for each round. Compared to the aggregation rule as in Eq. (3), where a small $p^\tau_{s|(i,b)}$ can greatly increase the global step size and dominate the update direction via $\frac{d_{i,s}G^\tau_{(i,b),s}}{B_i p^\tau_{s|(i,b)}}$, Eq. (22) adjusts this term to $\frac{d_{i,s}(G^\tau_{(i,b),s} - h^\tau_{i,s})}{B_i p^\tau_{s|(i,b)}}$ and includes stale updates from all clients (the first term of $\Delta_{\tau,s}$) for unbiased training. Given that $G^\tau_{(i,b),s}$ and $h^\tau_{i,s}$ should have similar update directions, the scale of $G^\tau_{(i,b),s} - h^\tau_{i,s}$ should be smaller than that of $G^\tau_{(i,b),s}$, mitigating the impact of small $p^\tau_{s|(i,b)}$ in the aggregation. However, despite these convergence benefits, this new aggregation rule requires extra memory at the server to record and maintain the stale updates $\{h^\tau_{i,s}\}$.

In experiments, we find that this aggregation rule greatly improves the performance of MMFL-GVR, due to its high heterogeneity of client participation from $\mathbf{p}^\tau$. Therefore, we use **MMFL-GVR*** to refer to the training algorithm that uses allocation probabilities $\mathbf{p}^\tau$ from Theorem 9 and the aggregation rule given in Eq. (22).

## VI. Model Fairness with Variance-Reduced Sampling

We finally introduce a variant of our sampling algorithms that addresses *model* heterogeneity in addition to client heterogeneity. Since different models may require varying rounds of training to converge, prior works have pointed out that fairness over model's performance (e.g., optimizing the minimum convergence time across tasks) is also a challenge [11]. Intuitively, one might wish to allocate more resources to the more "difficult" tasks, for faster training. [11] proposed allocating clients to tasks based on global loss values: $p_s = \frac{f_s^{\alpha-1}}{\sum_s f_s^{\alpha-1}}$, where $p_s$ is the probability of assigning a client to train model $s$, and $f_s^{\alpha-1}$ is the *alpha-fair* global loss of model $s$. Increasing $\alpha$ allocates more clients to poorly performing models.

However, this algorithm ignores client heterogeneity by assigning equal probabilities for all clients to any specific model ($p_{s|i} = p_{s|j}$). Therefore, we extend [11] by proposing a variant of our optimal **V**ariance-**R**educed client sampling algorithm (**MMFL-FairVR**) which solves the following problem:

$$\min_{\mathbf{p}^\tau} \sum_{s=1}^{S} \mathbb{E}[(\sum_{(i,b)\in\mathcal{A}_{\tau,s}} P^\tau_{(i,b),s} f_{i,s}(w_s^\tau) - \sum_{i\in\mathcal{N}_s} d_{i,s} f_{i,s}(w_s^\tau))^2] \tag{23}$$

$$\text{s.t. } p^\tau_{s|(i,b)} > 0,\ \sum_{s\in\mathcal{S}_i} p^\tau_{s|(i,b)} \le 1,\ \sum_{i\in\mathcal{N}_s}\sum_{b=1}^{B_i} p^\tau_{s|(i,b)} = m_s,$$

[5] With this rule, the server must maintain $h^\tau_{i,s}$ for all models and clients.

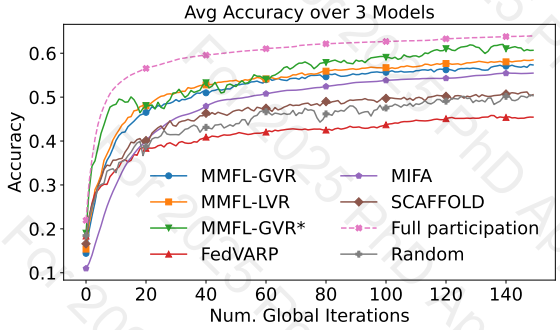


Fig. 2. Average accuracy of different methods. The experiment includes 3 models. MMFL-GVR, MMFL-LVR, MMFL-GVR* achieves higher accuracy compared to other baselines. MMFL-GVR* performs the best, achieving 96% of the final accuracy reached by the full participation benchmark, even though MMFL-GVR* only assumes 10% of the processors participate in each round.

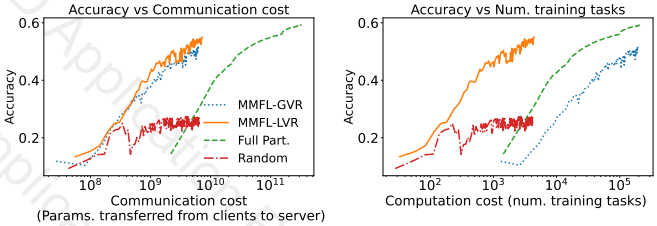


Fig. 3. Comparison of sampling methods by cost metrics (1 random seed). Left: Accuracy vs. Communication Cost. Right: Accuracy vs. Local Computation Cost. The x-axis represents the accumulated communication/computation cost from the first round to the current round. *Full participation* achieves the highest accuracy but incurs high communication and computation costs. MMFL-GVR performs similarly to MMFL-LVR in terms of accuracy but with significantly higher computation costs.

where $m_s = mp_s$ is the expected number of processors allocated to model $s$, with $p_s = \frac{f_s^{\alpha-1}}{\sum_s f_s^{\alpha-1}}$, i.e. $m_s$ is determined by the prevailing relative performance levels across models. Aside from this constraint, Eq. (23)'s optimization problem is the same as MMFL-LVR's in Eq. (16). Given the amount of training resources (processors) per model $m_s$, we solve Eq. (23) for $\mathbf{p}^\tau$, the probability distribution of processor $(i, b)$ being allocated to model $s$ in round $\tau$. Since Eq. (23) is a convex optimization problem, it can be solved with any convex solver. Intuitively, we prioritize selecting clients whose data distributions have not been so well represented during training yet, for better model convergence and less bias in the heterogeneous data distribution scenario, whilst taking into account model fairness.

## VII. Experiment and Evaluation

In this section, firstly we present the evaluation of our proposed algorithms, MMFL-GVR, MMFL-LVR, MMFL-GVR*, and MMFL-FairVR, against established benchmarks. We use as benchmarks FedVARP [30], MIFA [31], SCAFFOLD [32], *Random* (where $\mathbf{p}^\tau$ follows a uniform distribution), and *full participation*, which represents the theoretically best performance under given local training settings in the FL framework. Since we construct an MMFL system with heterogeneous computational abilities, which is not addressed by previous MMFL works [7]–[15], their allocation algorithms cannot be applied in this scenario. The first three benchmarks are originally designed for single-model FL with modifications on local training (SCAFFOLD) or aggregation (FedVARP and MIFA) to improve single-model FL performance (global accuracy). For the client sampling process, SCAFFOLD and FedVARP originally used combinatorial selection ($N$ choose $m$), and MIFA adopted independent sampling with a distribution specified by its experimental settings. For a fair comparison, we adjust their sampling processes to align with our client-model assignment process using a uniform distribution.

Our experiments first *compare MMFL-GVR and MMFL-LVR against our baselines, and then to each other*. We find that MMFL-GVR generates highly heterogeneous assignment distribution $\mathbf{p}^\tau$, impacting convergence. MMFL-GVR* shows significantly better convergence, even outperforming MMFL-LVR. We then *assess MMFL-FairVR against FedFairMMFL [11] and Random*, showing that MMFL-FairVR leads to higher minimum and average accuracy across the models.

### A. Experimental Setup

We use four datasets for our experiments: Fashion-MNIST, EMNIST, CIFAR-10, and Shakespeare [1]. We construct 120 clients for all experiments. For the first three datasets, to simulate data heterogeneity, each client receives data from 30% of the labels. To further increase data heterogeneity across multiple models, clients are divided into two groups for each model: high-data clients (comprising 10% of the total clients, each holding around 120 datapoints for the respective model) and low-data clients (the remaining 90%, each holding around 12 datapoints for the respective model). Importantly, this division is model-specific, meaning a client can be a high-data client for one model and a low-data client for another model. Consequently, 10% of the clients hold approximately 52.6% of the total data for each model. The Shakespeare dataset is naturally non-iid, so we randomly select 120 clients uniformly from its total 1146 clients (each corresponding to a Shakespeare character) without any modifications. We use $m = 12$, so that 10% of clients are active in expectation, for our algorithms, and we assume each client is active with 10% probability for the other baselines. The number of local training epochs is set to $E = 5$ for all models. For the Fashion-MNIST and EMNIST classification tasks, we construct similar convolutional neural networks (CNN), each with 2 convolutional layers, 2 pooling layers, and 2 linear layers, and with different output layer sizes. For the CIFAR-10 task, we use a pre-activation ResNet [33]. For the Shakespeare dataset, we implement a character-level LSTM language model with an embedding layer, a two-layer LSTM, and a linear layer. All algorithms are implemented in Pytorch 2.2.1 with `SGD optimizer` [34]. Experiments are performed and results averaged over **5 random seeds**.

We have $S$ models, and assume some clients may be unable to train specific models: 90% of clients can train all $S$ models,

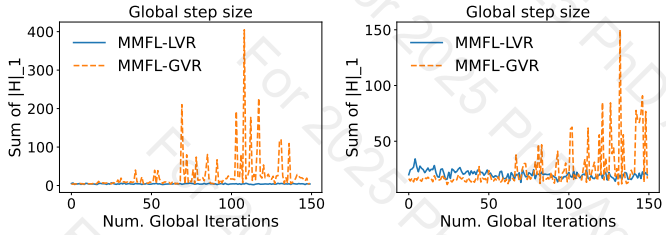


Fig. 4. Comparison of the summed global step size of all models ($\sum_{s=1}^{S}\|H_{\tau,s}\|_1 = \sum_{s=1}^{S}\sum_{(i,b)\in\mathcal{A}_{\tau,s}} P^{\tau}_{(i,b),s}$) Left: 3-model setting. Right: 5-model setting. MMFL-GVR's global step size is much more unstable, potentially harming the training stability. In contrast, MMFL-LVR's participation heterogeneity is much lower, leading to more stable convergence.

while 10% can train $S-1$ models (randomly decided). Each client $i$ has $B_i$ processors, allowing them to train $B_i$ models in parallel per round. Clients' $B_i$ distribution is as follows: 25% of clients have $B_i = |\mathcal{S}_i|$ (the number of models available for client $i$), 50% have $B_i = \lceil \frac{|\mathcal{S}_i|}{2} \rceil$, and 25% have $B_i = 1$.

### B. Experiment Results

*1) 3-Model Setting:* We first construct an MMFL system with three Fashion-MNIST models. The results, shown in Fig. 2, indicate that our proposed methods (MMFL-LVR, MMFL-GVR, and MMFL-GVR*) outperform baselines, with final accuracies close to that of the *full participation* scenario (theoretically the best under the same local training settings).

**Comparison of MMFL-GVR and MMFL-LVR.** MMFL-LVR, which uses loss-based sampling, achieves slightly better performance than MMFL-GVR (gradient-based sampling), with a 2.13% higher average final accuracy across models. Figure 3 further compares these two methods with the *full participation* and *random* baselines in terms of **communication and computation costs**. *Full participation* achieves higher accuracy but requires all clients to communicate with the server per round, which can be unaffordable for the server. Although MMFL-GVR performs similarly to MMFL-LVR, it requires all clients to train all models and upload gradient norms to generate $\mathbf{p}^\tau$, resulting in much higher computation cost (Fig. 3 right). In contrast, MMFL-LVR only requests clients to compute loss values, which is more affordable on the client-side, making it more feasible for real applications.

**Reason for MMFL-GVR's Slightly Lower Accuracy.** The sampling distribution $\mathbf{p}^\tau$ generated by MMFL-GVR is more unbalanced between processors, increasing the heterogeneity of client participation. This is because local loss values are typically within a specific range, whereas gradient norms can vary significantly across clients. As a result, MMFL-GVR's $\mathbf{p}^\tau$ is more heterogeneous, leading to an unstable global step size, as discussed in Section V. Figure 4 shows the total global step size ($\sum_{s=1}^{S}\|H_{\tau,s}\|_1$) for these two methods. It is clear that MMFL-GVR's global step size is more unstable. As seen in Fig. 2, MMFL-GVR* improves upon the original MMFL-GVR by including stale updates in the aggregation.

*2) 5-Model Setting:* To further challenge our algorithms, we increased the task heterogeneity by training 5 models on different datasets: two Fashion-MNIST, one CIFAR-10, one EMNIST, and one Shakespeare. The quantitative results for this setting and the previous 3-model setting is in Table I. We report the relative accuracy compared to that of full participation (i.e., each method's average final accuracy divided by the average final accuracy under full participation). Since we use the same client participation rate as in the 3-model setting, each model's training resource is more limited when spread over five models, leading to a performance drop shown in the table for all methods. Consistent with Fig. 2's results, MMFL-LVR and MMFL-GVR* come closest to the average final model accuracy with full participation.

TABLE I
FINAL AVERAGE MODEL ACCURACY RELATIVE TO THAT FROM FULL PARTICIPATION (THEORETICALLY THE BEST UNDER THE SAME LOCAL TRAINING SETTINGS).

| Methods | 3 tasks | 5 tasks | Comm. Cost | Comp. Cost | Mem. Cost |
|---|---|---|---|---|---|
| FedVARP [30] | 0.712±.14 | 0.690±.19 | Low | Low | High |
| MIFA [31] | 0.868±.18 | 0.835±.18 | Low | Low | High |
| SCAFFOLD [32] | 0.794±.14 | 0.650±.24 | Low | Low | Low |
| Random | 0.778±.19 | 0.749±.23 | Low | Low | Low |
| Full Participation | 1.000±.13 | 1.000±.14 | High | High | Low |
| MMFL-GVR | 0.893±.14 | 0.842±.20 | Low | High | Low |
| MMFL-LVR | 0.912±.15 | 0.849±.16 | Low | Low | Low |
| MMFL-GVR* | **0.960**±.15 | **0.869**±.18 | Low | High | High |

Table I also compares the communication, computation, and extra memory costs of all methods. *Full participation* requires all processors to train all models, resulting in high communication and computation costs. Compared to MMFL-LVR and other baselines, MMFL-GVR and MMFL-GVR* incur high computation costs as all clients must train all models to obtain the gradient norm. In contrast, MMFL-LVR only requires clients to update their loss values, resulting in lower computation costs. FedVARP [30], MIFA [31], and MMFL-GVR* all require extra memory to store stale updates.

*3) Model Fairness Experiment:* In Fig. 5, we finally evaluate our MMFL-FairVR in the same 5-model setting described above. We compare our MMFL-FairVR against FedFairMMFL [11], which constructs $p_s$ (the probability of assigning a client to train model $s$) with $\alpha = 3$ based on global loss values. By considering client heterogeneity, our MMFL-FairVR achieves approximately 3.70% higher minimum accuracy across models compared to FedFairMMFL, and much better than *random*; MMFL-FairVR and FedFairMMFL both have comparable average final accuracies across models. By solving Eq. (23), we minimize the variance of the training, leading to a higher minimum accuracy across all models.

## VIII. CONCLUSION

In this work, we consider a MMFL system with heterogeneous computation abilities and available models at each FL client. We first analyze the convergence in this setting when each client has a given probability of training each model. Motivated by the goal of reducing our convergence

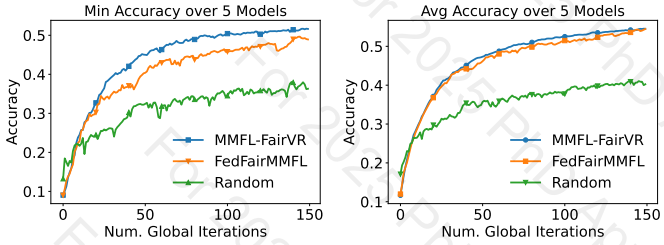


Fig. 5. Model fairness experiment. Left: minimum accuracy across all models. Right: average accuracy across all models. The sampling distribution of MMFL-FairVR is well optimized given the heterogeneous clients and tasks, achieving a much higher minimum accuracy and comparable average accuracy across models compared to other baselines.

upper bound, we then propose MMFL-GVR and MMFL-LVR, which respectively assign clients to models so as to minimize the variance of the gradient norm and loss across all clients assigned to a given model. We also address the participation heterogeneity caused by different client participation rates, an overlooked factor in most of the past literature. To mitigate the impact of high participation heterogeneity during training, we provide an aggregation rule with stale updates (MMFL-GVR*). Additionally, we introduce MMFL-FairVR to address model fairness in MMFL and improve the performance of the poorly performing models within the system. All proposed algorithms were validated through extensive experiments under various settings, demonstrating their performance advantages.

## 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 *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] 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.

[3] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

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

[5] 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.

[6] M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *arXiv preprint arXiv:1901.09888*, 2019.

[7] B. Askin, P. Sharma, C. Joe-Wong, and G. Joshi, "Fedast: Federated asynchronous simultaneous training," in *Uncertainty in artificial intelligence*. Proceedings of the Uncertainty in Artificial Intelligence Conference, 2024.

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

[9] N. Bhuyan, S. Moharir, and G. Joshi, "Multi-model federated learning with provable guarantees," in *EAI International Conference on Performance Evaluation Methodologies and Tools*. Springer, 2022, pp. 207–222.

[10] 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.

[11] M. Siew *et al.*, "Fair concurrent training of multiple models in federated learning," *arXiv:2404.13841*, 2024.

[12] S. K. Atapour, S. J. Seyedmohammadi, S. M. Sheikholeslami, J. Abouei, A. Mohammadi, and K. N. Plataniotis, "Multi-model federated learning optimization based on multi-agent reinforcement learning," in *2023 IEEE 9th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2023, pp. 151–155.

[13] J. Liu, J. Jia, B. Ma, C. Zhou, J. Zhou, Y. Zhou, H. Dai, and D. Dou, "Multi-job intelligent scheduling with cross-device federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 535–551, 2022.

[14] Z.-L. Chang, S. Hosseinalipour, M. Chiang, and C. G. Brinton, "Asynchronous multi-model dynamic federated learning over wireless networks: Theory, modeling, and optimization," *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[15] J. Liu, H. Cao, A. S. M. Tayeen, S. Misra, P. Kumar, and J. Harikumar, "Multi-model-based federated learning to overcome local class imbalance issues," in *2023 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023, pp. 265–270.

[16] G. Lan, X.-Y. Liu, Y. Zhang, and X. Wang, "Communication-efficient federated learning for resource-constrained edge devices," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 1, pp. 210–224, 2023.

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

[18] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[19] Z. Zhao and G. Joshi, "A dynamic reweighting strategy for fair federated learning," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8772–8776.

[20] 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," in *44th IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2024.

[21] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *IEEE INFOCOM 2022-IEEE conference on computer communications*. IEEE, 2022, pp. 1739–1748.

[22] 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.

[23] E. Rizk, S. Vlaski, and A. H. Sayed, "Optimal importance sampling for federated learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3095–3099.

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

[25] 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.

[26] Y. Ruan, X. Zhang, and C. Joe-Wong, "How valuable is your data? optimizing client recruitment in federated learning," *IEEE/ACM Transactions on Networking*, 2024.

[27] Y. J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 10 351–10 375.

[28] H. Sedrakyan and N. Sedrakyan, *Algebraic inequalities*. Springer, 2018.

[29] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards flexible device participation in federated learning for non-iid data," *arXiv preprint arXiv:2006.06954*, 2020.

[30] 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.

[31] 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.

[32] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learn-

ing," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.
[33] 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.
[34] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.