



# Hardware-Sensitive Fairness in Heterogeneous Federated Learning

ZAHIDUR TALUKDER, The University of Texas at Arlington, Arlington, United States

BINGQIAN LU, University of California Riverside, Riverside, United States

SHAOLEI REN, University of California Riverside, Riverside, United States

MOHAMMAD ATIQUIL ISLAM, Computer Science and Engineering, The University of Texas at Arlington, Arlington, United States

---

Federated learning (FL) is a promising technique for decentralized privacy-preserving Machine Learning (ML) with a diverse pool of participating devices with varying device capabilities. However, existing approaches to handle such heterogeneous environments do not consider “fairness” in model aggregation, resulting in significant performance variation among devices. Meanwhile, prior works on FL fairness remain hardware-oblivious and cannot be applied directly without severe performance penalties. To address this issue, we propose a novel hardware-sensitive FL method called FairHetero that promotes fairness among heterogeneous federated clients. Our approach offers tunable fairness within a group of devices with the same ML architecture as well as across different groups with heterogeneous models. Our evaluation under MNIST, FEMNIST, CIFAR10, and SHAKESPEARE datasets demonstrates that FairHetero can reduce variance among participating clients’ test loss compared to the existing state-of-the-art techniques, resulting in increased overall performance.

CCS Concepts: • **Security and privacy**;

Additional Key Words and Phrases: Federated Learning, Fairness, Hardware, Privacy

## ACM Reference Format:

Zahidur Talukder, Bingqian Lu, Shaolei Ren, and Mohammad Atiqul Islam. 2025. Hardware-Sensitive Fairness in Heterogeneous Federated Learning. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 10, 1, Article 5 (March 2025), 31 pages. <https://doi.org/10.1145/3703627>

---

## 1 Introduction

### 1.1 Motivation

In the wake of exploding user-generated data and the proliferation of **Machine Learning (ML)** and AI in our everyday lives, **federated learning (FL)** has emerged as a promising technique for distributed, collaborative, and privacy-preserving ML training across many devices. In FL, devices

---

Zahidur Talukder and Bingqian Lu contributed equally to this research.

Authors’ Contact Information: Zahidur Talukder, The University of Texas at Arlington, Arlington, Texas, United States; e-mail: zahidurrahim.talukder@mavs.uta.edu; Bingqian Lu, University of California Riverside, Riverside, California, United States; e-mail: blu029@ucr.edu; Shaolei Ren, University of California Riverside, Riverside, California, United States; e-mail: shaolei@ucr.edu; Mohammad Atiqul Islam, Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas, United States; e-mail: mislam@uta.edu.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2025 Copyright held by the owner/author(s).

ACM 2376-3639/2025/03-ART5

<https://doi.org/10.1145/3703627>

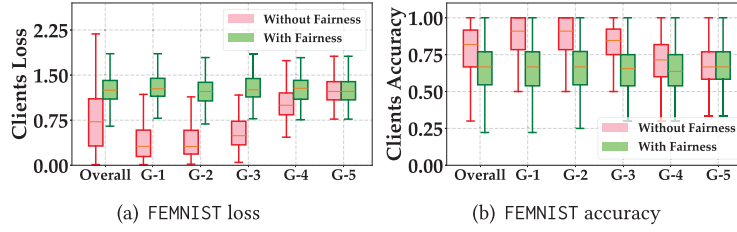


Fig. 1. The impact of imposing existing fairness approaches on FL clients with different hardware capabilities. Without fairness constraints, the clients can adopt different model architectures, and the client with the lowest architecture (G-5), where G-1 to G-5 represent decreasing model architecture sizes and limited hardware capability, suffers from inferior performance due to its smaller model. With fairness constraints, all clients are forced to adopt the smallest architecture, leading to severe performance degradation. Hence, existing approaches for performance fairness result in hardware unfairness as clients with better hardware do not perform better.

perform ML model training locally and send the model updates to a central server for aggregation [15, 16, 28]. These early works on FL force all devices to adopt identical ML models (i.e., homogeneous model architecture) for local training, even when different participating devices have different hardware capabilities. Meanwhile, in the pursuit of performance improvement, increasingly complex and specialized ML models are being developed and deployed, pushing devices to their computation limits [10, 31, 33]. With such progression in the ML model complexity, it has become impractical to restrict FL to homogeneous model architecture, which is limited by the weakest participating device. Consequently, new FL approaches are introduced, which allow devices to undertake ML model complexities in line with their hardware capabilities [8, 9, 20, 39]. However, heterogeneous FL exacerbates the performance disparity among devices as the distribution of device-level data and the model updates may vary significantly among different devices, leading to non-uniform performance among devices on the final trained model [13, 22, 23, 34–37]. These performance variations are undesirable as these “unfairly” advantage or disadvantage some devices. *Our goal in this article is to systematically rectify such performance bias and improve FL “fairness.”*

## 1.2 Limitations of Prior Works

Fairness in ML has garnered significant attention in recent years, with recent works in federated settings trying to address the performance disparity among FL clients [11, 23, 27, 40]. However, prior works suffer from two fundamental limitations. *First*, these approaches force every device to have the same model architecture, leading to significant degradation of overall performance due to the architecture bottleneck of the weakest hardware/device. We illustrate this in Figure 1 where imposing fairness (forcing the same model architecture for all) on heterogeneous devices causes severe performance loss. Note here that while fairness constraints typically cause some performance loss (mainly for clients with better performance without fairness), the performance degradation in Figure 1 is mostly due to enforcing architecture homogeneity. *Second*, existing FL fairness approaches typically focus on reducing the performance gap among clients by boosting the performance of those who are lagging behind. This approach assumes that all clients should achieve similar performance levels. However, in practical FL scenarios, it is expected that clients with better hardware, capable of running more complex ML architectures, will naturally outperform those with less capable devices [9]. While it is important to support weaker clients,

prioritizing them exclusively can introduce fairness concerns for other clients who are better equipped.

In our design, we do not aim for all clients to achieve the same or similar performance solely due to their hardware capabilities. Instead, we strive to reduce the overall variance in performance across clients, ensuring that disparities are minimized while acknowledging inherent hardware differences. This approach moves beyond traditional FL fairness models, addressing the need for “*hardware-sensitive*” fairness that better accommodates the diverse capabilities of clients in heterogeneous environments.

### 1.3 Our Contributions

In this article, we propose a novel hardware-sensitive framework, FairHetero, for FL with heterogeneous hardware and heterogeneous model architecture. The core of our solution is the separation of FL clients’ performance variation due to data heterogeneity and hardware heterogeneity. FairHetero first divides participating devices into groups based on their hardware (and model architecture) capabilities. It then applies a novel layered reweighting of the global objective function to enforce performance fairness. Moreover, FairHetero allows tuning the degree of reweighting for fairness—from no performance fairness to forcing every client to have uniform performance. At the group level, FairHetero imposes performance fairness constraints among the members of the same group, and we call it “intra-group fairness.” At the global level, FairHetero imposes performance fairness constraints among different groups, and we call it “inter-group fairness.” The intra-group reweighting tackles fairness due to data heterogeneity, and the inter-group reweighting tackles fairness due to hardware heterogeneity. By separately handling the two types of performance heterogeneity, FairHetero *allows a graceful and hardware-sensitive tradeoff between performance and fairness among heterogeneous devices*. Note here that FairHetero is not to be taken as a “naturally fair” FL approach; rather, it should be taken as a federation mechanism that allows fairness tuning at the data and hardware levels separately. Through this separation of fairness tuning, we can adopt more mainstream data fairness (i.e., reducing performance gaps among FL clients with heterogeneous data) along with hardware fairness (i.e., allowing some performance gaps among FL clients with heterogeneous model architecture). To the best of our knowledge, FairHetero algorithm is the first attempt to address fairness in FL with devices/clients with hardware heterogeneity. Our method ensures a more balanced performance among participating clients with different computational capabilities.

To evaluate FairHetero, we conduct a theoretical analysis to demonstrate that FairHetero reduces the performance variation of participating clients with different hardware capabilities. We also run extensive FL simulations using four datasets, MNIST, CIFAR10, FEMNIST, and SHAKESPEARE, under IID and Non-IID data distribution among FL clients. We analyze the impacts of our algorithm’s design parameters on FL performance and fairness, demonstrating the tuning capability offered by FairHetero. We compare our algorithm with state-of-the-art algorithms, showing that, in most cases, our algorithm enhances fairness among participating clients while improving overall performance.

The remainder of this article is organized as follows: Section 2 provides an overview of related works in the field of architectural heterogeneous FL and fairness-aware FL algorithms. Section 3 introduces the preliminaries and background concepts necessary to understand our approach. In Section 4, we present the design of FairHetero and discuss its key components with theoretical analysis, uniformity analysis, and generalization bound. Section 5 describes the evaluation settings, including datasets, experimental setup, and performance metrics used to evaluate our algorithm. In Section 6, we present the results of our experiments and compare FairHetero with state-of-the-art algorithms. Finally, Section 7 concludes the article and discusses future research directions.

## 2 Related Work

In this section, we review existing literature related to addressing heterogeneous model architecture and fairness in FL. We first discuss approaches that handle heterogeneous model architectures, including partial training, personalization, and knowledge distillation techniques. Subsequently, we explore methods that focus on achieving fairness in FL, highlighting their strategies and limitations.

### 2.1 Heterogeneous Model Architecture

To address the heterogeneous computational power of local clients, various heterogeneous model training for FL has been proposed. One line of work is based on **partial training (PT)** on different clients based on the computation power. For example, in Federated Dropout [3] heterogeneous model architecture is achieved by random dropout from the global model. In HeteroFL [9], a different subset of the model is trained at local devices based on their hardware capability. Another work FjORD [13] was introduced to tackle model heterogeneity by ordered dropout. FedRolex [1] takes the heterogeneous model architecture on a rolling basis, which solves the uneven training of the model parameters by the local dataset. Another type of work is based on the *personalization* model for the clients. Like in FedMask [20], personalized models are proposed with heterogeneous masking. In contrast, in Dispfl [8], the personalized model is extracted based on decentralized sparse training of the local clients. Besides, Fedhm [39] uses heterogeneous models via low-rank factorization. Some other previous works utilize *knowledge distillation*. FedDF [25] trains the classifier from the clients and uses a public dataset to further train the student model at the server. DSFL [14] uses an unlabeled public dataset at the server to train the student model by distillation-based semi-supervised learning. In FedET [4] a weighted distillation-based approach is proposed where the server model is trained on top of a smaller model on the clients.

### 2.2 Fairness in Federated Learning

Most existing work in FL focuses on homogeneous setups, where all clients are assumed to have similar hardware capabilities and model architectures. Fairness is a critical consideration in machine learning, and various methods have been developed to enhance fairness in FL, often tailored to specific application contexts [7, 30, 38]. As FL evolves, it introduces new challenges and different notions of fairness. A common approach in prior work defines fairness as the reduction of bias [6, 29], aiming to ensure that model outputs do not disproportionately favor or disadvantage certain groups defined by sensitive attributes. Another line of work defines fairness in terms of reducing performance variation caused by data heterogeneity. For instance, some approaches address data heterogeneity by forming distinct groups and ensuring fairness at the group level [11, 24]. Techniques like q-FFL [23] introduce a powered loss function with a parameter  $q$  before model aggregation, while AFL [12] uses a value function on the client side to select clients for the next iteration based on loss evaluations. Power-of-choice selection strategies [5] extend AFL by selecting clients with higher losses for subsequent training phases. These methods primarily focus on mitigating data heterogeneity among clients with homogeneous model architectures. Other approaches, such as Ditto [21] and CFFL [26], seek to improve client performance and achieve fairness through personalized models and collaborative learning.

In contrast, our work specifically addresses the performance disparities among clients resulting from differences in hardware capabilities. We aim to minimize the overall performance variation across clients by considering hardware heterogeneity as a crucial factor in achieving fairness. Existing methods do not explicitly address the challenges posed by hardware heterogeneity in FL, which our approach seeks to address.

### 3 Preliminaries

In this section, we provide an overview of the fundamental concepts related to our work. We start by defining FL and its optimization objective. We then discuss FL in the context of heterogeneous model architecture, where devices may have different computational capabilities and model sizes. Finally, we introduce the concept of fairness in FL and define our notion of performance fairness, which aims to ensure balanced performance across devices with varying hardware and data characteristics.

#### 3.1 Federated Learning

The goal of FL can be formulated as the following optimization problem:

$$\underset{\theta}{\text{minimize}} \quad \sum_{i=1}^N p_i f_i(\theta), \quad (1)$$

where  $N$  is the total number of devices while  $f_i(\theta)$  and  $p_i > 0$  are the local objective and weight parameter of device  $i$ , respectively. The typical choice of local objective  $f_i(\theta)$  is the empirical risk over the local dataset  $\mathcal{D}_i$ , i.e.,  $f_i(\theta) = \frac{1}{|\mathcal{D}_i|} \sum_{(x,y) \in \mathcal{D}_i} l(\theta, x, y)$ . We can set  $p_i = \frac{|\mathcal{D}_i|}{\sum_i |\mathcal{D}_i|}$  to achieve the minimum empirical risk over the entire data set across all devices. The solution of (1) in prior works (particularly federated averaging or FedAVG) involves communication-efficient update, where a subset of all devices apply stochastic gradient descent on their local dataset for multiple epochs before sending it to the aggregation server [28].

#### 3.2 FL with Heterogeneous Architecture

Instead of a homogeneously shared model, Reference [9] proposes to utilize heterogeneous models where each device trains a model appropriate to its own device capability. The key idea here is that weaker devices get a smaller model that can be nested within the centralized larger model.

Let us consider that  $N$  devices in FL are divided into  $M$  groups of devices, each group with  $N_m$  members sharing the same model architecture. Groups  $m$ 's model architecture,  $\theta_m$  is extracted from the centralized model as  $\theta \odot A_m$ , where  $A_m$  is a matrix with the same dimension as  $\theta$  consisting of only 0's and 1's, serving as a mask applied to the global model to obtain group  $m$ 's local training parameter  $\theta_m$ .  $\theta_m$  is a matrix of the same dimension as  $\theta$ , with 0's at positions outside of its desired model architecture  $A_m$ . Then for client  $i$  in group  $m$ , its loss function can be written as  $f_{m,i} = f_i(\theta_m) = f_i(\theta \odot A_m)$ .

We can update (1) as follows to incorporate the architecture heterogeneity:

$$\underset{\theta}{\text{minimize}} \quad \sum_{m=1}^M \sum_{i=1}^{N_m} p_{m,i} f_{m,i}(\theta \odot A_m). \quad (2)$$

We adopt the objective mentioned above for our design, utilizing the HeteroFL architecture [9] to categorize FL clients based on hardware capabilities. During aggregation, we use regions denoted by  $\mathcal{M}^{(j)}$  contributed by different subsets of model groups. Our objective can also be applied to other PT-based algorithms, such as FedRolex [1], with slight modifications. It is important to ensure that clients within the same group train the same subset of model parameters in each training round.

#### 3.3 Fairness

An FL system that solves (2) can introduce performance variation among devices due to their heterogeneity in model architecture and data. For instance, the central model will be biased toward devices with larger models and more data. Data creates performance variation among the devices

in the same architecture group, and we call this *intra-group performance variation*. Meanwhile, the architecture causes performance variation among different groups, and we call this *inter-group performance variation*. In this work, we seek to improve overall “performance fairness” in FL with heterogeneous architecture and define fairness as follows.

*Definition 3.1 (Intra-group fairness- Data Fairness).* For any hardware group  $m$  with  $N_m$  members trained with model  $\theta_m = (\theta \odot A_m)$  and  $\theta'_m = (\theta' \odot A_m)$ , we call  $\theta_m$  is more fair if

$$\text{Var}(f_{m,1}(\theta_m), \dots, f_{m,N_m}(\theta_m)) < \text{Var}(f_{m,1}(\theta'_m), \dots, f_{m,N_m}(\theta'_m)).$$

*Definition 3.2 (Inter-group fairness- Hardware Fairness).* For total  $M$  hardware groups trained with global model  $\theta$  and  $\theta'$ , we call  $\theta$  is more fair if

$$\text{Var}\left(\frac{1}{N_1} \sum_{i=1}^{N_1} f_{1,i}(\theta_1), \dots, \frac{1}{N_M} \sum_{i=1}^{N_M} f_{M,i}(\theta_M)\right) < \text{Var}\left(\frac{1}{N_1} \sum_{i=1}^{N_1} f_{1,i}(\theta'_1), \dots, \frac{1}{N_M} \sum_{i=1}^{N_M} f_{M,i}(\theta'_M)\right),$$

where  $\theta_m$  corresponds to the model architecture for group  $m$ .

Our definition of performance fairness is based on the uniformity of clients’ test loss on local data. While we formalize fairness using variance, other uniformity metrics, such as the accuracy distribution of clients and cosine similarity [23], can also be used without any loss of generality of our definition. Also, note that while we define inter-group fairness (Hardware Fairness) as having less performance variation among different hardware groups, our goal is to allow separate tuning capabilities to control the two types of fairness above. In the process, we become aware of hardware capability differences (allowing better hardware to perform better) instead of imposing a blanket performance fairness goal.

#### 4 Hardware Sensitive Fairness: FairHetero

In this section, we provide a detailed explanation of FairHetero’s objective and solution, along with theoretical analyses on its convergence, uniformity, and generalization bounds.

##### 4.1 Objective of FairHetero

To impose the fairness condition on (2), we reweight the objective function to favor the devices with higher loss by giving them higher weights. Our solution is inspired by  $\alpha$ -fairness in wireless networks and  $q$ -fairness for FL with homogeneous architecture [18, 23]. More specifically, we define the objective of our Fair FL with heterogeneous architecture (FairHetero) as

$$\underset{\theta}{\text{minimize}} \quad \sum_{m=1}^M \frac{g_m}{q+1} \left( \overbrace{\sum_{i=1}^{N_m} p_{m,i} f_{m,i}(\theta \odot A_m)^{q_m+1}}^{\text{Intra-Group Fairness}} \right)^{\frac{q+1}{q_m+1}}. \quad (3)$$

Here  $q$  and  $q_m$  are hyperparameters for tuning the inter-group and intra-group fairness, respectively.  $g_m$  is the group weight meeting the condition  $\sum_m g_m = 1$ . We achieve both intra-group and inter-group fairness in (3) by employing a layered weighting approach. The hyperparameter  $q_m$  is used to minimize variance in the losses among devices within a group  $m$ , which shares the architecture mask  $A_m$ . By adjusting  $q_m$ , we control intra-group fairness, with each group potentially having a unique value of  $q_m$ . To minimize variance across different groups, we use a global parameter  $q$ , which applies uniformly to all groups. In general, larger values of  $q$  and  $q_m$  enforce stricter fairness requirements. Figure 2 demonstrates the implementation of FairHetero. When there is

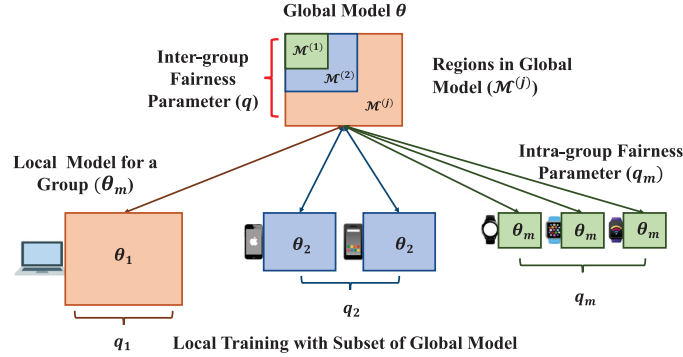


Fig. 2. The figure illustrates the architecture of a partial training-based heterogeneous setup in FL, where clients are grouped based on their hardware capabilities to train subsets of a global model. Region  $\mathcal{M}^{(1)}$  is shared across all client groups  $\theta_1$ ,  $\theta_2$ , and  $\theta_3$ . Region  $\mathcal{M}^{(2)}$  is shared between groups  $\theta_1$  and  $\theta_2$ , while region  $\mathcal{M}^{(j)}$  is exclusive to group  $\theta_1$ . The FairHetero algorithm introduces parameters  $q_m$  and  $q$  to ensure fair performance both within and across these federated client groups.

only one group (i.e.,  $M = 1$ ), we set  $g_1$  to 1,  $q$  to 0, and remove the normalization term  $\frac{1}{q_{m+1}}$ , since there is no inter-group fairness, then the former formulation is equivalent as the formulation of  $q$ -fairness [23].

**Necessity of layered weighting.** The model heterogeneity introduces additional performance variance, and naively applying global fairness as in prior work will result in significant performance degradation among devices with larger architecture and lower loss. By separating the performance variance due to architecture (i.e., inter-group variance) from performance variance due to data (i.e., intra-group variance), we allow a graceful implementation of fairness. Note that (3) is a generalized version of FL fairness and, therefore, also applicable to homogeneous architecture (i.e.,  $M = 1$ ).

**Tuning hyperparameters  $q$  and  $q_m$ .** The impact of certain values of  $q$  and  $q_m$  on the FL clients' performance distribution depends on the client datasets, loss function, and model architecture, making it impossible to “directly determine” the values of  $q$  and  $q_m$  for a certain level of fairness. Hence, FairHetero requires hyperparameter tuning.

## 4.2 Solution of FairHetero

We adopt a communication-efficient FL approach where, in each iteration, a device  $i$  in group  $m$  trains its masked model  $\theta \odot A_m$ . The device then sends its loss  $f_{m,i}$  and gradient  $\nabla_{\theta} f_{m,i} \odot A_m$  back to the central server. In scenarios where the server cannot determine the clients' capabilities or group affiliations, it can send the global model directly to the client, allowing the client to select a suitable subset of the model architecture. Our design imposes no restrictions on the number of groups, so a group may consist of a single client. Additionally, our approach does not require a coordinator for each group, unlike hierarchical designs.

The calculation of the group gradient and the norm of the group Hessian is essential for the FairHetero algorithm due to the heterogeneous architectures in federated learning settings. These calculations determine each group's contribution to the global model update, ensuring that updates from different groups are appropriately weighted based on their impact on the global model.

The calculation of group gradient and norm of group Hessian is necessary for the FairHetero algorithm because of the heterogeneous architecture in federated learning settings. The group gradient and norm of group Hessian are used to calculate the contribution of each group to the global

model update. Specifically, the norm of the Hessian is crucial for estimating the local Lipschitz constant of the gradient in FairHetero. More details are provided in Lemmas 1 and 2. This estimation allows for dynamic adjustment of the step size in gradient-based optimization methods, ensuring stable convergence without the need for manual tuning for each  $q$  and  $q_m$  value. By providing an upper bound on how much the gradient can change, the Hessian norm helps maintain efficiency and balance between accuracy and fairness across different  $q$  and  $q_m$  settings. This calculation ensures that updates from different groups are appropriately weighted based on their impact on the global model.

**Calculation of group gradient ( $\Delta_m$ ).** The group gradient regarding global model parameter  $\theta$  is as follows:

$$\begin{aligned}
& \nabla_{\theta} \left\{ \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_m+1}(\theta \odot A_m) \right)^{\frac{q+1}{q_m+1}} \right\} \\
&= \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} \{f_i(\theta \odot A_m)\} \\
&= \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \\
&= \Delta_m.
\end{aligned}$$

**Calculation of norm of group Hessian ( $H_m$ ).** The Hessian regarding global model parameter  $\theta$  is as follows:

$$\begin{aligned}
& \nabla_{\theta}^2 \left\{ \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_m+1}(\theta \odot A_m) \right)^{\frac{q+1}{q_m+1}} \right\} \\
&= \nabla_{\theta} \left\{ \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right\} \\
&= \frac{g_m}{q_m+1} \nabla_{\theta} \left\{ \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right. \\
&\quad \left. + \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \nabla_{\theta} \left\{ \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right\} \right\}.
\end{aligned} \tag{4}$$

For gradient in the first term, we have the following:

$$\begin{aligned}
& \nabla_{\theta} \left\{ \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \right\} \\
&= \frac{q-q_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-2q_m-1}{q_m+1}} \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m.
\end{aligned} \tag{5}$$

For gradient in the second term, we have the following:

$$\begin{aligned} & \nabla_{\theta} \left\{ \sum_{i=1}^{N_m} p_{m,i} (q_m + 1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right\} \\ &= \sum_{i=1}^{N_m} \left[ p_{m,i} (q_m + 1) q_m f_{m,i}^{q_m-1} (\nabla_{\theta} f_{m,i} \odot A_m) (\nabla_{\theta} f_{m,i} \odot A_m)^T + p_{m,i} (q_m + 1) f_{m,i}^{q_m} \nabla_{\theta}^2 f_{m,i} \odot A_m \right]. \end{aligned} \quad (6)$$

Plug the two equations above into Equation (4) as follows:

$$\begin{aligned} & \nabla_{\theta}^2 \left\{ \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_m+1} (\theta \odot A_m) \right)^{\frac{q+1}{q_m+1}} \right\} \\ &= \frac{g_m}{q_m+1} \frac{q-q_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-2q_m-1}{q_m+1}} \\ & \quad \left( \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right) \left( \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right)^T \\ & \quad + \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \\ & \quad \sum_{i=1}^{N_m} \left[ p_{m,i} (q_m+1) q_m f_{m,i}^{q_m-1} (\nabla_{\theta} f_{m,i} \odot A_m) (\nabla_{\theta} f_{m,i} \odot A_m)^T + p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta}^2 f_{m,i} \odot A_m \right]. \end{aligned} \quad (7)$$

For term  $(\sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m) (\sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m)^T$ ,  $(\nabla_{\theta} f_{m,i} \odot A_m) (\nabla_{\theta} f_{m,i} \odot A_m)^T$ , and  $\nabla_{\theta}^2 f_{m,i} \odot A_m$  terms in the Hessian above, we have the following:

$$\begin{aligned} & \left( \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right) \left( \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right)^T \\ & \leq \left\| \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right\|^2 \times I \end{aligned} \quad (8)$$

and

$$(\nabla_{\theta} f_{m,i} \odot A_m) (\nabla_{\theta} f_{m,i} \odot A_m)^T \leq \|\nabla_{\theta} f_{m,i} \odot A_m\|^2 \times I. \quad (9)$$

Suppose the non-negative function  $f(\cdot)$  has a Lipschitz gradient with constant  $L$ ,

$$\nabla_{\theta}^2 f_{m,i} \odot A_m \leq L \times I. \quad (10)$$

Plugging the three inequalities above into Equation (7), we have the following:

$$\begin{aligned}
& \nabla_{\theta}^2 \left\{ \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_m+1}(\theta \odot A_m) \right)^{\frac{q+1}{q_m+1}} \right\} \\
& \leq \frac{g_m}{q_m+1} \frac{q-q_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-2q_m-1}{q_m+1}} \left\| \sum_{i=1}^{N_m} p_{m,i}(q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right\|^2 \times I \\
& \quad + \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \\
& \quad \sum_{i=1}^{N_m} \left[ p_{m,i}(q_m+1) q_m f_{m,i}^{q_m-1} \|\nabla_{\theta} f_{m,i} \odot A_m\|^2 \times I + p_{m,i}(q_m+1) f_{m,i}^{q_m} L \times I \right].
\end{aligned} \tag{11}$$

Therefore, the norm of the group Hessian ( $H_m$ ) can be written as follows:

$$\begin{aligned}
& \left\| \nabla_{\theta}^2 \left\{ \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_m+1}(\theta \odot A_m) \right)^{\frac{q+1}{q_m+1}} \right\} \right\| \\
& \leq \frac{g_m}{q_m+1} \frac{q-q_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-2q_m-1}{q_m+1}} \left\| \sum_{i=1}^{N_m} p_{m,i}(q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right\|^2 \\
& \quad + \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1} \right)^{\frac{q-q_m}{q_m+1}} \\
& \quad \sum_{i=1}^{N_m} \left[ p_{m,i}(q_m+1) q_m f_{m,i}^{q_m-1} \|\nabla_{\theta} f_{m,i} \odot A_m\|^2 + p_{m,i}(q_m+1) f_{m,i}^{q_m} L \right] \\
& = H_m.
\end{aligned} \tag{12}$$

**Adjusting the learning rate** A critical concern in tuning hyperparameters  $q$  and  $q_m$  is the adjustment of the learning rate  $\gamma$  for each pair value. This adjustment becomes particularly challenging for gradient-based methods, where the step size is inversely proportional to the Lipschitz constant of the function's gradient. Changing the values of  $q$  and  $q_m$  can lead to significant fluctuations in the step size, potentially causing training instability or divergence.

To address this issue, we propose estimating the local Lipschitz constant for FairHetero's objective by tuning the step size on just one value of  $q$  and  $q_m$  (e.g.,  $q = 0$  and  $q_m = 0$ ). This approach allows us to dynamically adjust the step size for our gradient-based optimization method without the need for manual tuning of  $q$  and  $q_m$ . The norm of the Hessian is used to estimate the local and global Lipschitz constant of the gradient, which in turn helps in determining an appropriate step size for gradient-based optimization methods when solving FairHetero objectives.

**LEMMA 1 (UPPER BOUND FOR GROUP LEVEL HESSIANS).** *If the non-negative function  $f(\cdot)$  has a Lipschitz gradient with constant  $L$ , then for any  $q_m \geq 0$  and any point  $\theta$ ,*

$$\left\| \nabla_{\theta}^2 \left\{ p_{m,i} f_i^{q_m+1}(\theta \odot A_m) \right\} \right\| \leq p_{m,i}(q_m+1) q_m f_{m,i}^{q_m-1} \|\nabla_{\theta} f_{m,i} \odot A_m\|^2 + p_{m,i}(q_m+1) f_{m,i}^{q_m} L = L_{gp}, \tag{13}$$

is an upper bound for the local Lipschitz constant of the gradient of  $p_{m,i}f_i^{q_{m+1}}(\theta \odot A_m)$  at point  $\theta$ . Refer to the Appendix in supplementary materials for the complete proof of Lemma 1.

LEMMA 2 (UPPER BOUND FOR GLOBAL LEVEL HESSIANS). *If the non-negative function  $F(\cdot)$  has a Lipschitz gradient with constant  $L$ , then for any  $q \geq 0$  and any point  $\theta$ ,*

$$\begin{aligned}
& \left\| \nabla_{\theta}^2 \left\{ \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_{m+1}}(\theta \odot A_m) \right)^{\frac{q+1}{q_{m+1}}} \right\} \right\| \\
& \leq \frac{g_m}{q_m+1} \frac{q-q_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_{m+1}} \right)^{\frac{q-2q_m-1}{q_{m+1}}} \left\| \sum_{i=1}^{N_m} p_{m,i} (q_m+1) f_{m,i}^{q_m} \nabla_{\theta} f_{m,i} \odot A_m \right\|^2 \\
& \quad + \frac{g_m}{q_m+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_{m+1}} \right)^{\frac{q-q_m}{q_{m+1}}} \\
& \quad \sum_{i=1}^{N_m} \left[ p_{m,i} (q_m+1) q_m f_{m,i}^{q_m-1} \|\nabla_{\theta} f_{m,i} \odot A_m\|^2 + p_{m,i} (q_m+1) f_{m,i}^{q_m} L \right] \\
& = L_G
\end{aligned} \tag{14}$$

is an upper bound for the group Lipschitz constant of the gradient of

$$\frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_{m+1}}(\theta \odot A_m) \right)^{\frac{q+1}{q_{m+1}}}$$

at point  $\theta$ . Refer to the Appendix in supplementary materials for the complete proof of Lemma 2.

**Remark 1. Fairness-performance tradeoff.** Increasing the value of  $q$  enhances the performance of smaller architecture clients. However, if  $q$  approaches  $+\infty$ , then it transforms into a max-min problem [32], favoring smaller architectures most at the expense of larger ones. Striking a balance is crucial, aiming for a slight decline in larger architecture performance to significantly boost smaller architecture clients, ensuring an overall improved average performance across all participants. Our experiments in Section 5 demonstrate that a lower  $q$  achieves greater uniformity among heterogeneous architecture clients.

**Model aggregation.** Due to heterogeneous architecture, we aggregate the models by dividing the global model  $\theta$  into non-overlapping regions, which have an equal number of devices contributing to model updates. For instance, in Figure 2, we have three regions in the global model  $\theta$ : the green region gets model updates from all devices, the blue region gets model updates from devices with architecture  $\theta_1$  and  $\theta_2$ , and the light-red region gets updates from only devices with  $\theta_1$  architecture.

Let us consider there are  $J$  regions in the global model. We denote all the groups that contain region  $j$ 's parameter (non-zero value in  $A_m$ ) as set  $\mathcal{M}^{(j)}$ . For a group  $m \in \mathcal{M}^{(j)}$ , its contribution to global model update is  $\frac{\Delta_m^{(j)}}{\sum_{m \in \mathcal{M}^{(j)}} H_m^{(j)}}$ , where  $\Delta_m^{(j)}$  and  $H_m^{(j)}$  denote the part of  $\Delta_m$  or  $H_m$  that belongs to region  $j$ . Finally, the global server updates the model parameter as

$$\theta_{r+1} = \theta_r - \frac{\sum_{m \in \mathcal{M}^{(j)}} \Delta_m^{(j)}}{\sum_{m \in \mathcal{M}^{(j)}} H_m^{(j)}}. \tag{15}$$

Our solution to (3), FairHetero, is summarized in Algorithm 1.

**ALGORITHM 1:** FairHetero

---

```

1: Input: Global model  $\theta$ , group mask  $A_m$ , number of FL iterations  $R$ , and learning rate  $\gamma$ 
2: Output: Optimal architecture and weight  $\theta^*$  for each group
3: Initialization: Initial model parameter  $\theta_0$ 
4: for each federated learning round  $r = 1, \dots, R$  do
5:   Server sends global model parameter  $\theta$  to all clients
6:   for each group  $m = 1, \dots, M$  in parallel do
7:     Get the desired local model architecture  $A_m$ 
8:     for each client  $i = 1, \dots, N_m$  in group  $m$  in parallel do
9:       Local trainable model parameter:  $\theta_m = \theta \odot A_m$ 
10:      Client local update:
11:      for each local epoch  $t = 1, \dots, T$  do
12:         $\theta_{m,t} = \theta_{m,t-1} - \gamma \nabla_{\theta} f_{m,i,t-1}$ 
13:      end for
14:      Local parameter update after  $T$  epochs:  $\nabla_{\theta} f_{m,i} \odot A_m = L(\theta_m - \theta_{m,T})$ 
15:      Each client sends loss  $f_{m,i}$  and gradient  $\nabla_{\theta} f_{m,i} \odot A_m$  to the central server
16:    end for
17:  end for
18:  Server global aggregation:
19:  for each region  $j = 1, \dots, J$  do
20:    Server updates  $\theta_{r+1} = \theta_r - \frac{\sum_{m \in \mathcal{M}^{(j)}} \Delta_m^{(j)}}{\sum_{m \in \mathcal{M}^{(j)}} H_m^{(j)}}$ 
21:  end for
22: end for

```

---

**4.3 Theoretical Analysis of FairHetero**

In this section, we provide convergence analysis, generalization bound, and uniformity analysis for FairHetero. For detailed theoretical analysis and proof of theorems and lemmas, refer to the Appendix in supplementary materials.

**ASSUMPTION 1 (SMOOTHNESS).** *Loss functions  $f_1, \dots, f_N$  are all  $L$ -smooth:  $\forall \theta, \phi \in \mathcal{R}^d$  and any client  $i$  from group  $m$ , we assume that there exists  $L > 0$ :*

$$\|\nabla_{\theta} f_i(\theta_m) - \nabla_{\phi} f_i(\phi_m)\| = \|\nabla_{\theta} f_i(\theta \odot A_m) - \nabla_{\phi} f_i(\phi \odot A_m)\| \leq L \|\theta \odot A_m - \phi \odot A_m\|.$$

**ASSUMPTION 2 (ARCHITECTURE SLICING-INDUCED NOISE).** *We assume that for some  $\delta \in [0, 1)$  and any round  $r$ , group  $m$  with desired architecture  $A_m$ , the architecture slicing-induced noise is bounded by:*

$$\|\theta - \theta \odot A_m\|^2 \leq \delta^2 \|\theta\|^2,$$

where  $\theta$  denotes the global model parameters in round  $r$ , and  $A_m$  is the desired model architecture for clients in group  $m$ .

**ASSUMPTION 3 (BOUNDED GRADIENT).** *The expected squared norm of stochastic gradients is bounded uniformly, i.e., for constant  $G > 0$  and any round  $r$ , client  $i$  from group  $m$ , and its local training epoch  $t$ :*

$$\mathbb{E} \|\nabla_{\theta} f_i(\theta_m, \xi_{m,i,t})\|^2 \leq G,$$

where  $\xi_{m,i,t}$  is the local training dataset for client  $i$  used in local training epoch  $t$  and  $\theta_m$  is the trainable model parameter for group  $m$ :  $\theta_m = \theta \odot A_m$ .

ASSUMPTION 4 (GRADIENT NOISE FOR IID DATA). *Under IID data distribution. for any round  $r$ , client  $i$  from group  $m$  and its local training epoch  $t$ , we assume that*

$$\begin{aligned}\mathbb{E}\|\nabla_{\theta} f_i(\theta_m, \xi_{m,i,t})\| &= \nabla_{\theta} F(\theta_m) \\ \mathbb{E}\|\nabla_{\theta} f_i(\theta_m, \xi_{m,i,t}) - \nabla_{\theta} F(\theta_m)\|^2 &\leq \sigma^2\end{aligned}$$

for constant  $\sigma > 0$  and independent samples  $\xi_{m,i,t}$ .

4.3.1 *Convergence Analysis.* For the convergence analysis, we show that the sum of the squared norm of the gradient of the global loss  $\nabla_{\theta} F(\theta)$  converges over  $R$  federated learning rounds. Specifically, in our formulation,

$$F(\theta) = \sum_{m=1}^M \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_m+1}(\theta \odot A_m) \right)^{\frac{q+1}{q_m+1}}. \quad (16)$$

THEOREM 4.1. *Under the Assumptions stated above, for a learning rate  $\gamma$  and  $R$  as total number of federated learning rounds and  $T$  as local training epochs, the upper bound for  $\frac{1}{R} \sum_{r=1}^R \mathbb{E}\|\nabla_{\theta} F(\theta_r)\|^2$ , where  $F(\cdot)$  is the global loss satisfies*

$$\begin{aligned}\frac{1}{R} \sum_{r=1}^R \mathbb{E}\|\nabla F(\theta_r)\|^2 &\leq \frac{1}{\sqrt{RT}} \mathbb{E}[F(\theta_0)] + \left( \frac{L^2 \sigma^2 N}{2RT\Gamma_{min}^*} + \frac{3L^3 \sigma^2 N}{2RT\Gamma_{min}^*} \right) + \sum_{r=1}^R \mathbb{E}\|\theta_r\|^2 \\ &\quad + \frac{L^2 NG}{2R\Gamma_{min}^*} + \frac{3LN\sigma^2}{2T\Gamma_{min}^*} + \frac{3L^3 \gamma \sqrt{T} NG}{2\sqrt{RT}\Gamma_{min}^*},\end{aligned}$$

where  $\Gamma_{min}^*$  is a composite coefficient with denominator  $|\mathcal{M}^{(j)}|_{min}$ . For detailed proof refer to the Appendix in supplementary materials.

This effectively demonstrates the convergence of our objective function after conducting  $R$  federated rounds. As mentioned earlier, FairHetero extends prior work in federated learning by offering a flexible tradeoff between performance and fairness through the parameterization of  $q$  and  $q_m$ .

4.3.2 *Generalization bound for FairHetero.* In this section, we first describe the setup of FairHetero in more details and then provide the generalization bound. One benefit of FairHetero is that it allows a flexible tradeoff between fairness and performance that generalizes to  $q$ -fairness (a special case when  $M = 1$ , we set  $g_1$  to 1,  $q$  to 0, and remove the normalization term  $\frac{1}{q_m+1}$ ). The generalization bound for FairHetero provides insights into its performance in federated learning scenarios with heterogeneous groups. The bound ensures that the empirical loss of the model, compared to its true loss, remains within a certain range with high probability.

#### Group level generalization:

Total loss of group  $m$  with unknown loss of each devices,

$$L_{\lambda_m}(\theta) = \sum_{i=1}^{N_m} \lambda_{m,i} \mathbb{E}_{(x,y) \sim D_{m,i}} l(\theta_m, (x, y)), \quad (17)$$

where  $\lambda_m$  is in a probability simplex  $\Delta_m$ ,  $N_m$  is the total number of clients in group  $m$ ,  $D_{m,i}$  is the local data distribution for client  $i$  from group  $m$ ,  $\theta_m$  is the local model parameter for clients in group  $m$ , and  $l$  is the loss. We use  $\hat{L}_{\lambda_m}$  as empirical loss,

$$\hat{L}_{\lambda_m}(\theta) = \sum_{i=1}^{N_m} \frac{\lambda_{m,i}}{n_i} \sum_{j=1}^{n_i} l(\theta_m, (x_{i,j}, y_{i,j})), \quad (18)$$

where  $n_i$  is the number of local data samples of client  $i$  from group  $m$  and  $(x_{i,j}, y_{i,j}) \sim D_{m,i}$ .

The objective of group  $m$  is

$$\min_{\theta_m} \sum_{i=1}^{N_m} p_{m,i} f_{m,i}^{q_m+1}; \quad (19)$$

consider an unweighted version of  $\min_{\theta_m} \sum_{i=1}^{N_m} f_{m,i}^{q_m+1}$ , which is equivalent to minimizing the empirical loss

$$\tilde{L}_{q_m}(\theta) = \max_{v_m, \|v_m\|_{p_m} \leq 1} \sum_{i=1}^{N_m} \frac{v_{m,i}}{n_i} \sum_{j=1}^{n_i} l(\theta_m, (x_{i,j}, y_{i,j})), \quad (20)$$

where  $\frac{1}{p_m} + \frac{1}{q_m+1} = 1$  ( $p_m \geq 1, q_m \geq 0$ ).

**LEMMA 3 (GENERALIZATION BOUND FOR A SPECIFIC  $\lambda_m$ ).** Assume that the loss  $l$  is bounded by  $Q > 0$  and the numbers of local samples are  $(n_1, \dots, n_{N_m})$ . Then for any  $\delta > 0$ , the following inequality holds with probability at least  $1-\sigma$  for any  $\lambda_m \in \Lambda_m, \theta \in \Theta$ :

$$L_{\lambda_m}(\theta) \leq A_{q_m}(\lambda_m) \tilde{L}_{q_m}(\theta) + \mathbb{E}[\max_{\theta \in \Theta} L_{\lambda_m}(\theta) - \hat{L}_{\lambda_m}(\theta)] + Q \sqrt{\sum_{i=1}^{N_m} \frac{\lambda_{m,i}^2}{2n_i} \log \frac{1}{\delta}}, \quad (21)$$

where  $A_{q_m}(\lambda_m) = \|\lambda_m\|_{p_m}$  and  $\frac{1}{p_m} + \frac{1}{q_m+1} = 1$ .

Refer to the Appendix in supplementary materials for the complete proof of Lemma 3.

**Generalization bound for any  $\lambda_m$ .** Assume that the loss  $l$  is bounded by  $Q > 0$  and the numbers of local samples are  $(n_1, \dots, n_{N_m})$ . Then for any  $\delta > 0$ , the following inequality holds with probability at least  $1-\sigma$  for any  $\lambda_m \in \Lambda_m, \theta \in \Theta$ :

$$L_{\lambda_m}(\theta) \leq \max_{\lambda_m \in \Lambda_m} (A_{q_m}(\lambda_m)) \tilde{L}_{q_m}(\theta) + \max_{\lambda_m \in \Lambda_m} \left( \mathbb{E}[\max_{\theta \in \Theta} L_{\lambda_m}(\theta) - \hat{L}_{\lambda_m}(\theta)] + Q \sqrt{\sum_{i=1}^{N_m} \frac{\lambda_{m,i}^2}{2n_i} \log \frac{1}{\delta}} \right), \quad (22)$$

where  $A_{q_m}(\lambda_m) = \|\lambda_m\|_{p_m}$ , and  $\frac{1}{p_m} + \frac{1}{q_m+1} = 1$ .

**Key takeaways.** For a given  $\lambda_m \in \Lambda_m$ , the bound ensures that the model's empirical loss is constrained by a combination of the weighted empirical loss and a term related to the distribution of local samples. This result highlights FairHetero's effectiveness in controlling the influence of each group's contribution to the overall loss, promoting fairness and balanced learning across different groups. Furthermore, for any  $\lambda_m \in \Lambda_m$ , the bound provides a broader guarantee by considering the maximum weighted empirical loss across all possible  $\lambda_m$ . This reinforces FairHetero's capacity to adapt to varying group characteristics, maintaining both fairness and performance in heterogeneous environments.

#### Global level generalization:

Total loss of all  $M$  groups (treat each group as a "client") with unknown loss of each groups,

$$L_{\lambda}(\theta) = \sum_{m=1}^M \lambda_m \mathbb{E}_{(x,y) \sim D_m} l(\theta_m, (x, y)), \quad (23)$$

where  $\lambda$  is in a probability simplex  $\Lambda$ ,  $M$  is the total number of groups,  $D_m$  is the local data distribution for group  $m$ ,  $\theta_m$  is the local model parameter for group  $m$ , and  $l$  is the loss. We define empirical loss  $\hat{L}_{\lambda}(\theta)$  as

$$\hat{L}_{\lambda}(\theta) = \sum_{m=1}^M \frac{\lambda_m}{n_m} \sum_{j=1}^{n_m} l(\theta_m, (x_{m,j}, y_{m,j})), \quad (24)$$

where  $n_m$  is the number of local data samples of group  $m$  and  $(x_{m,j}, y_{m,j}) \sim D_m$ .

The objective of federated learning over  $m$  groups is as follows:

$$\min_{\theta} \sum_{m=1}^M \frac{g_m}{q+1} \left( \sum_{i=1}^{N_m} p_{m,i} f_i^{q_{m+1}}(\theta \odot A_m) \right)^{\frac{q+1}{q_{m+1}}} = \min_{\theta} \sum_{m=1}^M \frac{g_m}{q+1} F_m^{\frac{q+1}{q_{m+1}}} = \min_{\theta} \sum_{m=1}^M \frac{g_m}{q+1} F_m'^{q+1}, \quad (25)$$

consider an unweighted version of  $\min_{\theta} \sum_{m=1}^M F_m'^{q+1}$ , which is equivalent to minimizing the empirical loss,

$$\tilde{L}_q(\theta) = \max_{v, \|v\|_p \leq 1} \sum_{m=1}^M \frac{v_m}{n_m} \sum_{j=1}^{n_m} l(\theta_m, (x_{m,j}, y_{m,j})), \quad (26)$$

where  $\frac{1}{p} + \frac{1}{q+1} = 1$  ( $p \geq 1, q \geq 0$ ).

**LEMMA 4 (GENERALIZATION BOUND FOR A SPECIFIC  $\lambda$ ).** Assume that the loss  $l$  is bounded by  $Q_{gp} > 0$  and the numbers of local samples are  $(n_1, \dots, n_M)$ . Then for any  $\delta > 0$ , the following inequality holds with probability at least  $1-\sigma$  for any  $\lambda \in \Lambda, \theta \in \Theta$ :

$$L_{\lambda}(\theta) \leq A_q(\lambda) \tilde{L}_q(\theta) + \mathbb{E}[\max_{\theta \in \Theta} L_{\lambda}(\theta) - \hat{L}_{\lambda}(\theta)] + Q_{gp} \sqrt{\sum_{m=1}^M \frac{\lambda_m^2}{2n_m} \log \frac{1}{\delta}}, \quad (27)$$

where where  $A_q(\lambda) = \|\lambda\|_p$ , and  $\frac{1}{p} + \frac{1}{q+1} = 1$ .

Refer to the Appendix in the supplementary materials for the complete proof of Lemma 4.

**Generalization bound for any  $\lambda$ .** Assume that the loss  $l$  is bounded by  $Q_{gp} > 0$  and the numbers of local samples are  $(n_1, \dots, n_M)$ . Then for any  $\delta > 0$ , the following inequality holds with probability at least  $1-\sigma$  for any  $\lambda \in \Lambda, \theta \in \Theta$ :

$$L_{\lambda}(\theta) \leq \max_{\lambda \in \Lambda} (A_q(\lambda)) \tilde{L}_q(\theta) + \max_{\lambda \in \Lambda} \left( \mathbb{E}[\max_{\theta \in \Theta} L_{\lambda}(\theta) - \hat{L}_{\lambda}(\theta)] + Q_{gp} \sqrt{\sum_{m=1}^M \frac{\lambda_m^2}{2n_m} \log \frac{1}{\delta}} \right), \quad (28)$$

where  $A_q(\lambda) = \|\lambda\|_p$ , and  $\frac{1}{p} + \frac{1}{q+1} = 1$ .

**Key takeaways.** The analysis extends to the global level by considering each group as a ‘‘client’’ in FL. The generalization bound on the total loss across all groups highlights FairHetero’s effectiveness in achieving fair and accurate models, even in the presence of diverse data distributions and varying group sizes.

In summary, the generalization bound for FairHetero highlights its ability to generalize well to unseen data while maintaining fairness and performance in federated learning settings with heterogeneous groups.

**4.3.3 Uniformity Induced by FairHetero.** In this section, we analyze the uniformity induced by FairHetero in both inter-group and intra-group contexts, as established by the convergence analysis. FairHetero can encourage more fair solution in terms of the entropy of the accuracy distribution (larger entropy). We begin by formally defining the notion of fairness in terms of entropy.

**Definition 4.2 (Intra-group: Entropy of Performance Distribution).** We say that the performance distribution for any hardware group  $m$  with  $N_m$  members  $\{(f_{m,1}(\theta_m), \dots, f_{m,N_m}(\theta_m))\}$  is more uniform under model parameter  $\theta$  than  $\theta'$  if

$$\tilde{H}(f(\theta)) \geq \tilde{H}(f(\theta')), \quad (29)$$

where  $\tilde{H}(f(\theta))$  is the entropy of the stochastic vector obtained by normalizing  $\{f_{m,1}(\theta_m), \dots, f_{m,N_m}(\theta_m)\}$ , defined as

$$\tilde{H}(f(\theta)) := - \sum_{i=1}^{N_m} \frac{f_{m,i}(\theta_m)}{\sum_{i=1}^{N_m} f_{m,i}(\theta_m)} \ln \left( \frac{f_{m,i}(\theta_m)}{\sum_{i=1}^{N_m} f_{m,i}(\theta_m)} \right). \quad (30)$$

*Definition 4.3 (Inter-group: Entropy of Performance Distribution).* For total  $M$  hardware groups  $\{(F_1(\theta_1), \dots, F_m(\theta_m))\}$  trained with global model  $\theta$  and  $\theta'$ , we call  $\theta$  is more fair if

$$\tilde{H}(F(\theta)) \geq \tilde{H}(F(\theta')), \quad (31)$$

where  $\tilde{H}(f(\theta))$  is the entropy of the stochastic vector obtained by normalizing  $\{(F_1(\theta_1), \dots, F_m(\theta_m))\}$ , defined as follows:

$$\tilde{H}(F(\theta)) := - \sum_{m=1}^M \frac{F_m(\theta_m)}{\sum_{m=1}^M F_m(\theta_m)} \ln \left( \frac{F_m(\theta_m)}{\sum_{m=1}^M F_m(\theta_m)} \right). \quad (32)$$

We next provide results based on Definitions 4.2 and 4.3. It states that for arbitrary  $q \geq 0$  and  $q_m \geq 0$ , by increasing  $q$  and  $q_m$  slightly, we can achieve more uniform performance distributions defined over higher orders of performance for both inter- and intra-group clients.

**Inter-group uniformity.** The objective function of FairHetero promotes inter-group uniformity, ensuring that each group's contribution to the overall loss is balanced. Mathematically, the unweighted version of the objective function can be expressed as

$$F(\theta) = \frac{1}{M} \sum_{m=1}^M \left( \frac{1}{N_m} \sum_{i=1}^{N_m} f_{m,i}^{q_m+1} \right)^{\frac{q+1}{q_m+1}} = \frac{1}{M} \sum_{m=1}^M F_m^{\frac{q+1}{q_m+1}}(\theta_m), \quad (33)$$

where  $F_m(\theta_m)$  is the sum of loss of group  $m$ :  $F_m(\theta_m) = \frac{1}{N_m} \sum_{i=1}^{N_m} f_{m,i}^{q_m+1}$ .

**LEMMA 5.** Let  $F(\theta)$  be twice differentiable in  $\theta$  with  $\nabla^2 F(\theta) > 0$  (positive definite). The derivative of  $\tilde{H}(F^{\frac{q+1}{q_m+1}}(\theta_q^*))|_{q=p}$  with respect to the variable  $q$  evaluated at the point  $q = p$  is non-negative, i.e.,

$$\frac{\partial}{\partial q} \tilde{H}\left(F^{\frac{q+1}{q_m+1}}(\theta_q^*)\right)|_{q=p} \geq 0, \quad (34)$$

where

$$\begin{aligned} \tilde{H}(F(\theta)) &:= - \sum_{m=1}^M \frac{F_m(\theta_m)}{\sum_{m=1}^M F_m(\theta_m)} \ln \left( \frac{F_m(\theta_m)}{\sum_{m=1}^M F_m(\theta_m)} \right) \\ \tilde{H}\left(F^{\frac{q+1}{q_m+1}}(\theta_q^*)\right) &:= - \sum_{m=1}^M \frac{F_m^{\frac{q+1}{q_m+1}}(\theta_{m,q}^*)}{\sum_{m=1}^M F_m^{\frac{q+1}{q_m+1}}(\theta_{m,q}^*)} \ln \left( \frac{F_m^{\frac{q+1}{q_m+1}}(\theta_{m,q}^*)}{\sum_{m=1}^M F_m^{\frac{q+1}{q_m+1}}(\theta_{m,q}^*)} \right). \end{aligned} \quad (35)$$

A complete proof of the Lemma 5 is provided in the Appendix in the supplementary materials.

where  $F_m(\theta_m)$  represents the loss of group  $m$ . The corresponding entropy term,  $\tilde{H}(F^{\frac{q+1}{q_m+1}}(\theta_q^*))$ , ensures that the distribution of losses across groups remains balanced. The proof shows that the derivative of this term with respect to  $q$  at  $q = p$  is non-negative, indicating that the objective promotes inter-group uniformity.

**Intra-group uniformity.** Similarly, FairHetero encourages intra-group uniformity by ensuring that each sample within a group contributes equally to the group's loss. The objective function for

Table 1. Dataset Description and Number of Clients for Different Datasets

Dataset	Training	Test	Clients Number	Distribution	Model
MNIST [19]	60,000	10,000	100	IID/Non-IID	MLP
CIFAR10 [17]	50,000	10,000	100	IID/Non-IID	CNN
FEMNIST [2]	341,873	40,832	3,383	Non-IID	MLP
SHAKESPEARE [2]	16,068	2,356	715	Non-IID	RNN

each group  $m$  is as follows:

$$F_m(\theta_m) = \frac{1}{N_m} \sum_{i=1}^{N_m} f_{m,i}^{q_m+1}. \quad (36)$$

LEMMA 6. Let  $F_m(\theta_m)$  be twice differentiable in  $\theta_m$  with  $\nabla^2 F_m(\theta) > 0$  (positive definite). The derivative of  $\tilde{H}(f^{q_m+1}(\theta_{q_m}^*))|_{q_m=p_m}$  with respect to the variable  $q_m$  evaluated at the point  $q_m = p_m$  is non-negative, i.e.,

$$\frac{\partial}{\partial q} \tilde{H}(f^{q_m+1}(\theta_{q_m}^*))|_{q_m=p_m} \geq 0, \quad (37)$$

where

$$\begin{aligned} \tilde{H}(f(\theta)) &:= - \sum_{i=1}^{N_m} \frac{f_{m,i}(\theta_m)}{\sum_{i=1}^{N_m} f_{m,i}(\theta_m)} \ln \left( \frac{f_{m,i}(\theta_m)}{\sum_{i=1}^{N_m} f_{m,i}(\theta_m)} \right) \\ \tilde{H}(f^{q_m+1}(\theta_{q_m}^*)) &:= - \sum_{i=1}^{N_m} \frac{f_{m,i}^{q_m+1}(\theta_{q_m}^*)}{\sum_{i=1}^{N_m} f_{m,i}^{q_m+1}(\theta_{q_m}^*)} \ln \left( \frac{f_{m,i}^{q_m+1}(\theta_{q_m}^*)}{\sum_{i=1}^{N_m} f_{m,i}^{q_m+1}(\theta_{q_m}^*)} \right). \end{aligned} \quad (38)$$

A complete proof of the Lemma 6 is provided in the Appendix in the supplementary materials.

The corresponding entropy term,  $\tilde{H}(f^{q_m+1}(\theta_{q_m}^*))$ , guarantees that the loss distribution within each group remains balanced. The proof establishes that the derivative of this term with respect to  $q_m$  at  $q_m = p_m$  is non-negative, demonstrating the promotion of intra-group uniformity.

**Algorithmic uniformity.** The algorithmic design of FairHetero ensures uniformity in both inter-group and intra-group contexts. By iteratively updating the model parameters to minimize the objective function while preserving the fairness constraints, FairHetero effectively balances the contributions of different groups and samples, thereby promoting uniformity in the federated learning process.

Overall, the uniformity induced by FairHetero plays a crucial role in ensuring fair and balanced federated learning across diverse and heterogeneous groups. A complete proof is provided in the Appendix in supplementary materials.

## 5 Evaluation Setup

This section discusses the experimental setup used to evaluate the performance of our algorithm. We first describe the datasets used and then detail the model parameters for each dataset, including the architectural diversity introduced to simulate different hardware capabilities among participating clients.

### 5.1 Dataset

We adopt four popular datasets MNIST, CIFAR10, FEMNIST, and SHAKESPEARE which are commonly used in literature [28].

**MNIST:** This dataset is well known for handwriting recognition and includes 70,000 grayscale images measuring 28×28 pixels. The dataset is split into 60,000 training samples and 10,000 test

samples. There are 10 different classes for the images, ranging from 0 to 9. We adopt three cases for MNIST data distribution for the clients i.e., IID, Non-IID, and Non-IID extreme. We distribute the training dataset evenly among 100 clients, with each client receiving 600 samples for IID cases. For the Non-IID cases we have one dominant class for each client having 80% of data and all other classes have the rest 20% data. Finally, for the Non-IID extreme cases, each class would have at most two classes of data. We also separate 10% of the client data for testing the model. The actual test set is also used to test the global performance of the model over time.

**CIFAR10:** Another popular dataset, CIFAR10 includes 60,000 colored images measuring 32x32 pixels. The dataset is divided into 50,000 training images and 10,000 test images, grouped into ten separate classes. Like the MNIST dataset, we have three types of data distribution for CIFAR10 i.e., IID, Non-IID, and Non-IID extreme. We divide the dataset into 100 clients, with each client receiving 500 samples for the IID cases. Similarly, for the Non-IID cases we have one dominant class for each client having 80% of data and all other classes have the rest 20% data. Finally, for the Non-IID extreme cases, each class would have at most two classes of data. We also separate 10% of the client’s data for the test of the model performance.

**FEMNIST:** The FEMNIST dataset, derived from the LEAF dataset and implemented in TensorFlow Federated, is divided among 3,383 unique users (we used the first 1000 users). It consists of 341,873 training examples and 40,832 test examples, featuring grayscale images measuring 28x28 pixels. The test set ensures representation from each user, creating a Non-IID (non-independent and identically distributed) and heterogeneous dataset. Each user represents a distinct client in this context. The test set from the distinct client ID is used for testing the model performance over time.

**SHAKESPEARE:** The SHAKESPEARE dataset is derived from *The Complete Works of William Shakespeare*. It utilizes the concept of speaking roles in plays to represent individual clients. The dataset comprises 715 genuine users (we used 71 clients with at least 60 test data points), with 16,068 training examples and 2,356 test examples in textual format. Similarly to FEMNIST, the test set includes at least one sample from each user. This dataset is also Non-IID and heterogeneous, with each user corresponding to a different client.

## 5.2 Model Parameters

We focus on an edge computing setup where our clients are IoT devices. Given the limited power and computational capacity of IoT devices, we opted for simpler, lightweight models. Instead of using actual hardware, we emulated the setup, creating different groups of clients to mimic varying hardware capabilities. The model parameters used in our training are detailed below.

**MNIST:** For the MNIST dataset, we employ a basic **multi-layer perceptron (MLP)** classifier using TensorFlow’s Keras sequential model. The MLP has two hidden layers with ReLU activation, comprising 200 and 100 neurons, respectively, followed by an output layer with 10 neurons and softmax activation. Before training, the input features are flattened, and the labels are one-hot encoded. We utilize the Adam optimizer with a learning rate of 0.001 for IID and 0.00012 for Non-IID and extreme cases and categorical cross-entropy as the loss function. Training is conducted for 300 epochs across various scenarios, including IID, Non-IID, and extreme cases. To create architectural diversity, we prune the global model into five distinct groups, denoted as Group 1 to Group 5, each with different performance levels. Each group is composed of 20 clients with a subset of the model’s parameters. Specifically, Group 1 retains 100% of the global model, Group 2 retains 70%, Group 3 retains 50%, Group 4 retains 25%, and Group 5 retains 12%. These groups represent different hardware capabilities among participating clients. The model parameters for each group in the MNIST dataset are detailed in Table 2.

**CIFAR10:** For the CIFAR10 dataset, we employ a straightforward **Convolutional Neural Network (CNN)** classifier using TensorFlow’s Keras. The CNN architecture comprises two sets of

Table 2. Number of Model Parameters for Different Model Architecture Groups (G-1 to G-5) for Different Datasets

Dataset	Learning Rate	Epochs	Number of Model Parameters (% of Global Model)				
			G-1	G-2	G-3	G-4	G-5
MNIST [19]	0.001	300	178,110 (100)	120,480 (70)	84,060 (50)	40,785 (25)	20,067 (12)
CIFAR10 [17]	5e-5	2,000	1,060,138 (100)	596,770 (75)	265,626 (50)	125,806 (35)	66,706 (25)
FEMNIST [2]	0.001	600	50,890 (100)	34,990 (70)	22,270 (45)	12,730 (25)	6,370 (12)
SHAKESPEARE [2]	0.001	400	4,048,470 (100)	2,452,054 (75)	1,248,854 (50)	714,474 (35)	438,870 (25)

convolutional layers, followed by a max-pooling layer, a dropout layer, and two fully connected layers with dropout regularization. ReLU is used as the activation function for the convolutional layers, while softmax is applied to the output layer. We utilize the “categorical-crossentropy” loss function, along with the Adam optimizer set to a learning rate of 0.001 for IID, and 0.00005 for Non-IID and extreme scenarios. The model is trained for 1,000 epochs for IID, and 2,000 epochs for Non-IID and extreme cases. To introduce architectural diversity, we vary the rate parameter in the model, which controls the number of neurons in the fully connected layers, resulting in five distinct architectures. Similarly to the MNIST setup, we create five architectural groups, each consisting of 20 clients. In this setup, Group 1 retains 100% of the model parameters, Group 2 retains 75%, Group 3 retains 50%, Group 4 retains 35%, and Group 5 retains 25%. These groups are designed to mimic different hardware capabilities among the participating clients. The model parameters for each architectural group for the CIFAR10 dataset are detailed in Table 2.

**FEMNIST:** For the FEMNIST dataset, we employ a simple MLP with two hidden layers, using fully connected dense layers with ReLU activation functions. The input shape of the model is 784, corresponding to the number of pixels in each image. The first hidden layer consists of 64 neurons, and the output layer contains 10 neurons without an activation function (as the loss function used is SparseCategoricalCrossentropy with from\_logits=True). The optimization process uses a learning rate of 0.001, without any regularization techniques applied. Training is conducted for 600 epochs to ensure convergence. To introduce architectural diversity, we prune the global model to create five distinct architectures with varying performances. Each group consists of 200 clients. Similarly to the setups for other datasets, Group 1 retains 100% of the model parameters, Group 2 retains 70%, Group 3 retains 45%, Group 4 retains 25%, and Group 5 retains 12%. These groups are designed to simulate different hardware capabilities among the participating clients. The model parameters for each group for the FEMNIST dataset are provided in Table 2.

**SHAKESPEARE:** For the SHAKESPEARE dataset, we implement a **Recurrent Neural Network (RNN)** using a GRU layer with stateful=True, ensuring the model’s state is maintained across batches. Input data is pre-processed using a lookup table that maps each ASCII character to an index, then segmented into sequences of length 50+1. The model includes an embedding layer with a batch input shape of [8, None], followed by a GRU layer, and concludes with a dense layer containing 86 output units. Training spans 400 epochs, with a custom function serving as the evaluation metric, measuring the accuracy of the model’s predictions across all characters in the input sequence. To introduce architectural diversity, we create five groups with 30 clients in each. Group 1 retains 100% of the model’s parameters, Group 2 retains 75%, Group 3 retains 50%, Group 4 retains 35%, and Group 5 retains 25%. These groups simulate varying hardware capabilities among the clients. The model parameters for each group for the SHAKESPEARE dataset are outlined in Table 2.

Our FairHetero approach is scalable with respect to the number of groups, as adding more groups does not increase computational overhead. By using client masking, it accommodates various model architectures efficiently. Additionally, FairHetero is adaptable to more advanced architectures as federated learning systems evolve. It remains effective for a range of models,

including deep CNNs, RNNs, and transformers, through architecture-specific adjustments like layerwise or attention-head pruning. This flexibility ensures that FairHetero can handle increased complexity and continue to minimize performance disparities even as model architectures become more sophisticated.

## 6 Results

In this section, we present the results of our experiments evaluating the performance of FairHetero. In the figure, G-1 corresponds to clients with the highest trainable model parameters, and G-5 corresponds to the lowest, in decreasing order, with the OVERALL column representing the performance of all clients together. We begin by analyzing the effect of the design parameter  $q_m$  on intra-group or data fairness, followed by the impact of  $q$  on inter-group or hardware fairness. We then discuss the overall performance of FairHetero across different datasets. Finally, we provide tabulated results for various datasets, showcasing the group-level variances and means for different values of  $q$  and  $q_m$ .

### 6.1 Effect of $q_m$ :- Intra-Group/Data Fairness

In our evaluation, we investigate the impact of the intra-group or data fairness metric ( $q_m$ ) on promoting fairness among clients within groups (similar hardware or same model architecture) for data heterogeneity. We maintain the inter-group or hardware fairness metric,  $q = 0$ , and vary the  $q_m$  value from 0 to 100 on the MNIST, CIFAR10, FEMNIST, and SHAKESPEARE datasets. The dataset consists of five groups based on Table 2, each representing different hardware characteristics. In our analysis, we keep  $q_m$  the same for all groups, i.e., for example, if  $q_m$  is set to 10, then it means that all groups have a  $q_m$  value of 10. Our findings found that increasing the  $q_m$  value reduces the variance within each group, as shown in Figure 3. However, there is a tradeoff between fairness and performance. With increasing  $q_m$ , we see a slight degradation of overall average performance. Notably, the client-level  $q_m$  metric cannot capture variance caused by architectural differences among client groups, leading to persistent performance disparities between groups due to hardware heterogeneity. Therefore, while the group-level  $q_m$  metric effectively reduces performance variance among clients within the same architectural groups due to data heterogeneity, it is unable to address hardware heterogeneity among different hardware groups.

**Key takeaways.** Increasing the intra-group/data fairness parameter ( $q_m$ ) reduces variance within groups but results in a slight degradation of overall average performance, highlighting a tradeoff between fairness and performance, particularly in addressing data heterogeneity within groups. It is unable to address performance variation due to hardware heterogeneity.

### 6.2 Effect of $q$ :- Inter-group/Hardware Fairness

To assess the impact of the inter-group or hardware fairness metric ( $q$ ) on reducing variance among groups with architectural heterogeneity, we conducted experiments using the MNIST, CIFAR10, and FEMNIST datasets. Five distinct groups were formed based on Table 2, each representing different hardware characteristics. By setting the intra-group or data fairness metric ( $q_m$ ) to 0, we varied the value of  $q$  from 0 to 500 for different datasets. Our findings reveal that increasing the value of  $q$  enhances model fairness by reducing variance among participating groups, particularly benefiting lower architectural client groups by improving their performance and reducing test loss leading to min-max problems.

Interestingly, we observed a trend where increasing the value of  $q$  up to a certain point leads to minimal degradation of larger architectural clients while improving the performance of smaller architectural clients, resulting in both fairer and improved performance. However, beyond a certain threshold, further increases in  $q$  can lead to divergence, causing the performance of all client

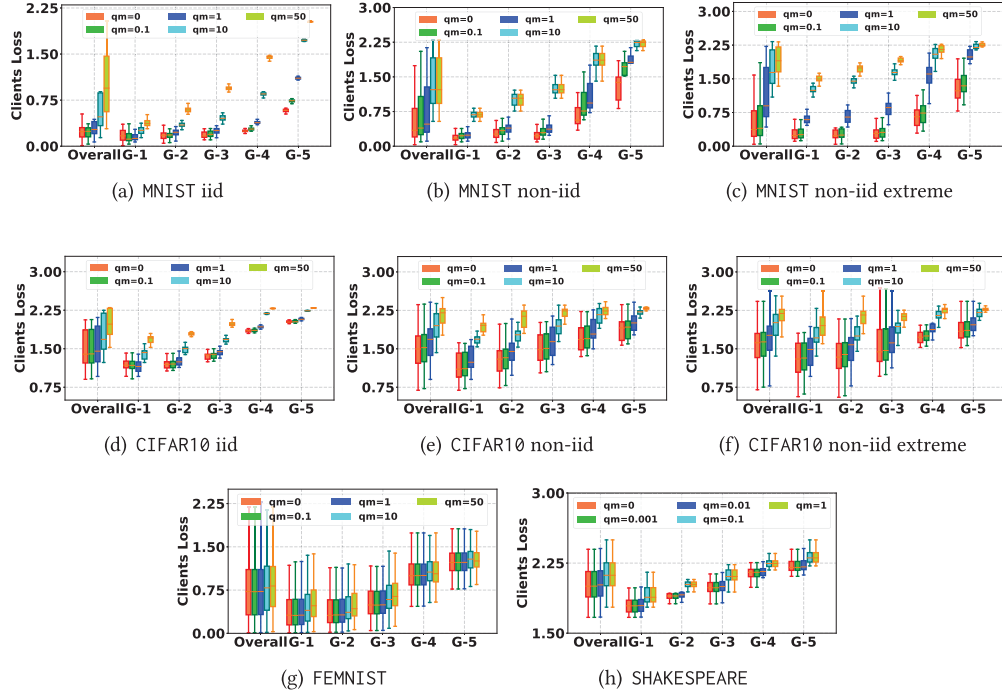


Fig. 3. Boxplot showing the effect of  $q_m$  on test loss for all datasets, with each box representing a group (G-1–G-5 from highest to lowest model architecture) and the overall box representing all clients. Each group contains multiple clients, and the boxplot provides a visual representation of the test loss distribution within and across groups.

groups to degrade despite reduced performance variance, ultimately resulting in a more equitable performance among groups. This highlights a tradeoff between performance and fairness, where increasing fairness too much may lead to performance loss.

Figure 4 illustrates that higher values of  $q$  decrease inter-group variance, while also reducing the overall performance as measured by the average test loss up to a certain  $q$  value, indicating an overall improvement in performance. However, beyond this optimal point, further increases in  $q$  lead to performance degradation. Thus, while the global fairness metric ( $q$ ) can effectively reduce inter-group variance among groups with architectural heterogeneity, it comes with a tradeoff in performance after a certain threshold of  $q$ .

**Key takeaways.** Increasing the inter-group fairness metric ( $q$ ) reduces variance among groups with different hardware capabilities, particularly benefiting less capable groups. However, this improvement is balanced by a tradeoff with performance, which can degrade after reaching a certain point. We found a wide spectrum of  $q$  values where it ensures both fairness and performance for all the datasets making it suitable for tuning.

### 6.3 Performance of FairHetero

FairHetero demonstrates a notable reduction in overall performance variance among clients with varying hardware computational power. In our experiments on four datasets (MNIST, FEMNIST, CIFAR10, and SHAKESPEARE) with architecturally heterogeneous groups, we compare FairHetero with existing partial training-based algorithms—HeteroFL [9], FedRolex [1] and FjORD [13] and fair q-FFL [23] algorithm, modified to address architectural heterogeneity—in terms of test loss

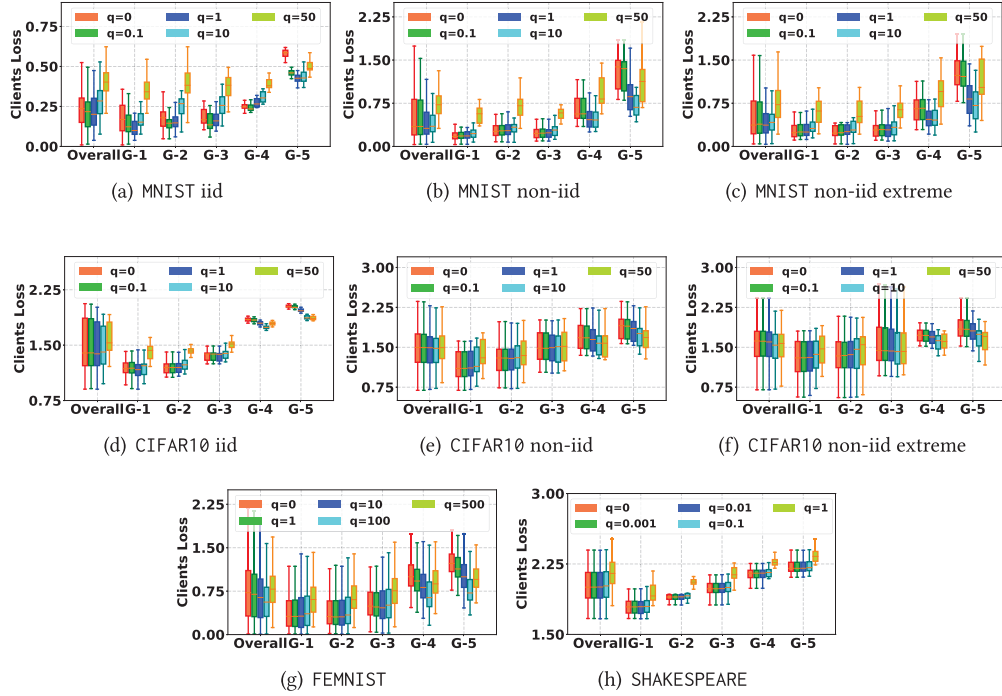


Fig. 4. Boxplot illustrating the impact of  $q$  on test loss for all datasets, where each box represents a group (G-1–G-5 from highest to lowest model architecture), and the overall box represents all clients. Each group contains multiple clients, and the boxplot visually depicts the test loss distribution within and across groups.

and test accuracy for each client, averaged across five random shuffles of each dataset. The results, as shown in Figure 5 for testing loss and Figure 6 for testing accuracy, indicate that with the tuning of the parameters  $q$  and  $q_m$  in Table 3, FairHetero achieves fairer solutions compared to existing methods. On average, FairHetero reduces the variance of test loss across all devices by up to 30% with improved average performance. In Table 3, we provide details on the worst and best 10% testing losses of the participating clients, as well as the variance of the final loss distributions. Comparing FairHetero with the existing algorithms, we observe that the proposed objective maintains similar or better average testing loss and accuracy while significantly reducing the variance of the performance of participating devices, ensuring uniformity.

**Key takeaways.** FairHetero demonstrates a significant reduction in overall performance variance among clients with varying hardware computational power compared to existing algorithms while maintaining similar or better average testing loss and accuracy, ensuring fairness and uniformity in performance.

#### 6.4 Comparison of Different Datasets with Varied $q$ and $q_m$ : Group-level Variance and Mean Analysis

A natural question comes as to how to choose the appropriate values of  $q$  and  $q_m$ . Our proposed FairHetero is flexible to tune  $q$  and  $q_m$  as tradeoffs between fairness (more uniformity of performance among participating clients) and overall performance (Average performance of all the clients) after a certain value of  $q$ . However, we found that we can attain a large degree of fairness among participating clients with overall better performance in most of the cases with smaller

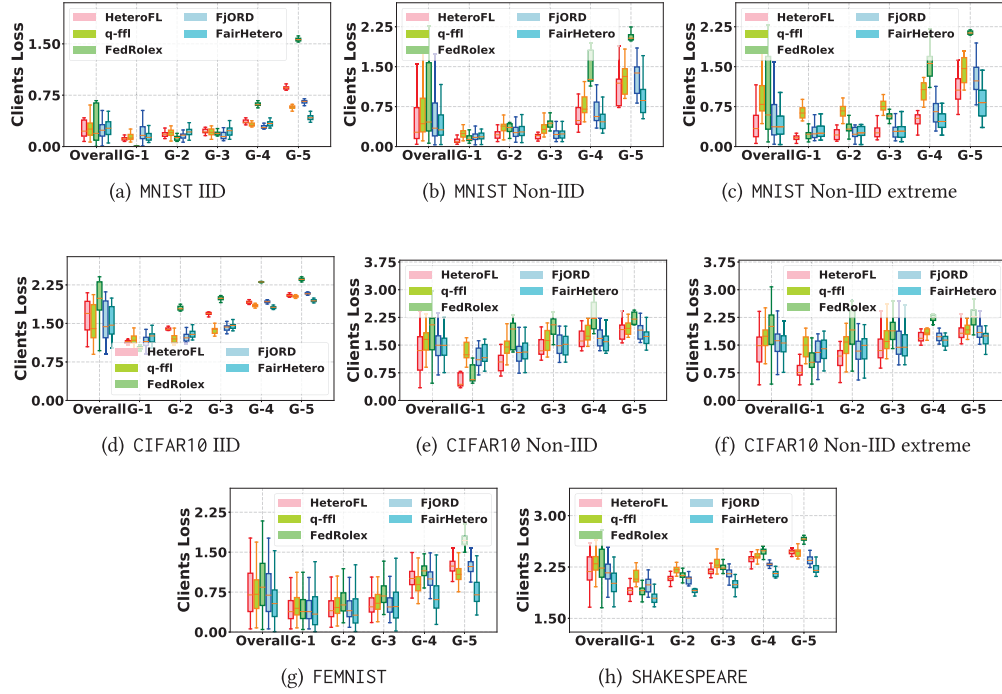


Fig. 5. FairHetero leads to fairer test loss distributions compared to the SOTA algorithms for IID and Non-IID data distribution for all the datasets.

values of  $q$  and  $q_m$ . Another important benefit of  $q_m$  is that we can tune the degree of fairness at the group level. Each group with a different architecture can have different  $q_m$ , so  $q$  and  $q_m$  can be tuned like hyperparameters based on the use cases and objectives.

We provide the detailed outcomes of our experiments where we tested various combinations of  $q$  and  $q_m$  values for the MNIST, CIFAR10, FEMNIST, and SHAKESPEARE datasets. The following table presents the results, illustrating how the distribution of clients' performance can be influenced by different values of  $q$  and  $q_m$ . In all the cases we found that added  $q$  and  $q_m$  can effectively reduce the variance of the performance of the clients having different architectures.

**Outcome of MNIST dataset evaluation.** Tables 4–6 present the average performance of each group in the MNIST IID, Non-IID, and extreme datasets, showing the mean and variance of the loss for different values of  $q$  and  $q_m$ . Across all groups, as we keep the  $q$  value fixed and increase the  $q_m$  value, we observe an increase in mean losses for all groups, but the variance of performance within each group decreases. This indicates that increasing  $q_m$  helps address the data heterogeneity within each group but does not address the hardware heterogeneity.

For increasing values of  $q$ , we notice a significant increase in the performance of the lowest architecture groups, with minimal degradation in performance for the highest architecture group's clients. This leads to a more uniform performance across different architecture groups. However, as the  $q$  value increases further, we observe an interesting trend. Instead of benefiting the performance of the lowest architectural clients, the performance of all clients starts to degrade, with the highest architecture clients suffering more. While this may result in a more balanced performance across the groups, it comes at the cost of average performance degradation, highlighting the tradeoff between fairness and performance.

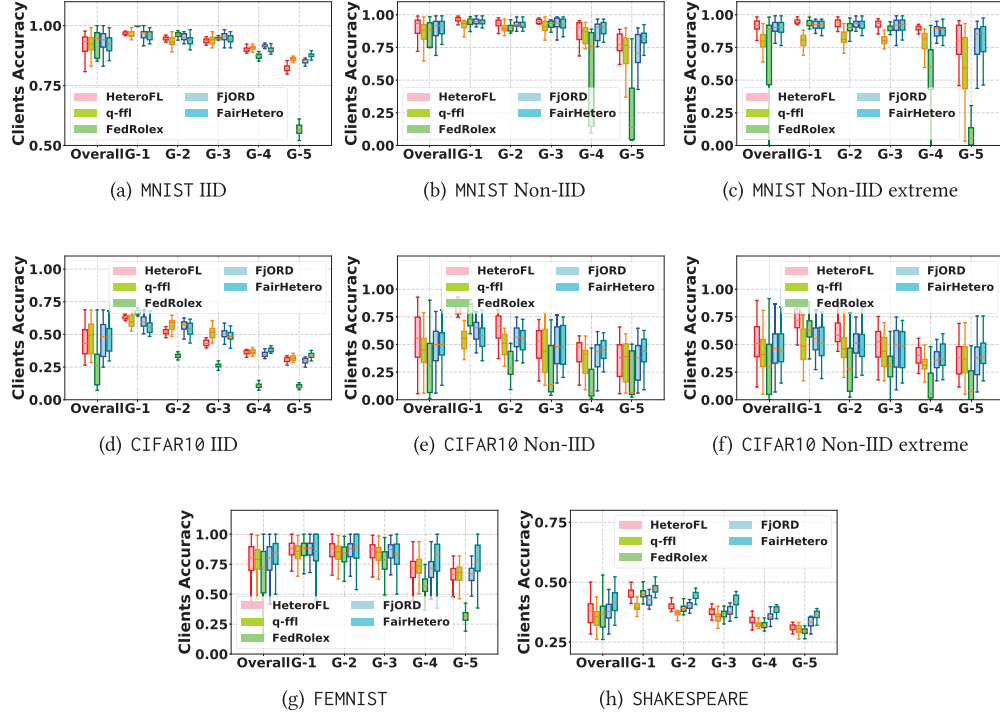


Fig. 6. FairHetero leads to fairer test accuracy distributions compared to SOTA algorithms for IID and Non-IID data distribution for all the datasets.

The optimal values of  $q$  for MNIST IID, MNIST Non-IID, and MNIST extreme are found to be in the range 1–10, 1–10, and 1–10, respectively. Across all groups, FairHetero has a greater impact on the performance of clients with limited resources, such as those in Group 5. Conversely, Group 1, representing clients with the highest computational power, is less affected by increasing  $q$  and  $q_m$ , indicating that these clients are less constrained by fairness considerations. The results demonstrate FairHetero’s ability to balance fairness and performance across a range of computational capabilities for MNIST dataset, making it effective in addressing heterogeneity in federated learning environments.

**Outcome of CIFAR10 dataset evaluation.** Tables 7–9 display the average performance of each group in the CIFAR10 IID, Non-IID, and extreme datasets, indicating the mean and variance of the loss for various  $q$  and  $q_m$  values. Increasing  $q_m$  reduces the variance within each group but does not address hardware heterogeneity. For increasing  $q$ , there is initially a significant improvement in the performance of lower architecture groups, but beyond a certain point, all groups experience performance degradation, with higher architecture groups suffering more. The optimal  $q$  values for CIFAR10 IID, Non-IID, and extreme datasets are in the range of 1–10. FairHetero has a greater impact on clients with limited resources (e.g., Group 5) but less on clients with higher computational power (e.g., Group 1). Overall, FairHetero effectively balances fairness and performance across various computational capabilities for the CIFAR10 dataset for the CNN model, addressing heterogeneity in federated learning environments.

**Outcome of FEMNIST dataset evaluation.** Table 10 presents the average performance of each group in the FEMNIST dataset, showing the mean and variance of the loss for different  $q$  and  $q_m$

Table 3. Comparison of Uniformity Measurements and Variance Analysis of FairHetero with HeteroFL, q-fll, FedRolex, and FjORD with Test Loss Considering Performances of All the Participating Clients from All Groups with Different Model Architectures

Dataset	Objective ( $q, qm_1 - qm_5$ )	Average	Worst 10%	Best 10%	Variance
MNIST IID	HeteroFL [9]	0.35 ± 0.27	0.86 ± 0.51	0.11 ± 0.0	0.07 ± 0.27
	q-fll [23]	0.29 ± 0.16	0.58 ± 0.29	0.11 ± 0.04	0.01 ± 0.12
	FedRolex [1]	0.50 ± 0.57	1.56 ± 1.06	0.00 ± 0.07	0.33 ± 0.57
	FjORD [13]	0.29 ± 0.20	0.66 ± 0.36	0.10 ± 0.00	0.04 ± 0.20
	FairHetero (10,0.1,0.1,0.1,0.1,0.1)	0.28 ± 0.11	<b>0.42 ± 0.14</b>	0.12 ± 0.05	<b>0.01 ± 0.11</b>
MNIST Non-IID	HeteroFL [9]	0.53 ± 0.46	1.38 ± 0.84	0.13 ± 0.06	0.21 ± 0.46
	q-fll [23]	0.61 ± 0.43	1.30 ± 0.69	0.18 ± 0.00	0.18 ± 0.43
	FedRolex [1]	0.90 ± 0.74	2.03 ± 1.14	0.14 ± 0.03	0.54 ± 0.74
	FjORD [13]	0.45 ± 0.42	1.18 ± 0.74	0.09 ± 0.06	0.18 ± 0.42
	FairHetero (1,0.001,0.001,0.001,0.001,0.001)	0.43 ± 0.33	<b>0.93 ± 0.50</b>	0.14 ± 0.04	<b>0.11 ± 0.33</b>
MNIST Extreme	HeteroFL [9]	0.56 ± 0.44	1.22 ± 0.66	0.15 ± 0.04	0.20 ± 0.44
	q-fll [23]	0.91 ± 0.34	1.43 ± 0.52	0.56 ± 0.01	0.12 ± 0.34
	FedRolex [1]	0.97 ± 0.76	2.14 ± 1.16	0.21 ± 0.01	0.57 ± 0.76
	FjORD [13]	0.47 ± 0.40	1.03 ± 0.56	0.13 ± 0.06	0.16 ± 0.40
	FairHetero (1,0.001,0.001,0.001,0.001,0.001)	0.45 ± 0.32	<b>0.81 ± 0.36</b>	0.16 ± 0.02	<b>0.10 ± 0.32</b>
CIFAR10 IID	HeteroFL [9]	1.64 ± 0.33	1.14 ± 0.16	2.04 ± 0.41	0.11 ± 0.33
	q-fll [23]	1.53 ± 0.36	2.02 ± 0.49	1.12 ± 0.05	0.13 ± 0.36
	FedRolex [1]	1.89 ± 0.48	<b>1.03 ± 0.38</b>	2.35 ± 0.46	0.23 ± 0.48
	FjORD [13]	1.56 ± 0.38	2.08 ± 0.51	1.13 ± 0.06	0.14 ± 0.38
	FairHetero (10,0.1,0.1,0.1,0.1,0.1)	1.55 ± 0.29	1.94 ± 0.39	1.22 ± 0.04	<b>0.08 ± 0.29</b>
CIFAR10 Non-IID	HeteroFL [9]	1.31 ± 0.54	1.99 ± 0.68	0.44 ± 0.32	0.30 ± 0.54
	q-fll [23]	11.64 ± 0.33	2.10 ± 0.46	1.19 ± 0.11	0.11 ± 0.33
	FedRolex [1]	1.80 ± 0.65	2.42 ± 0.62	0.57 ± 0.58	0.43 ± 0.65
	FjORD [13]	1.50 ± 0.38	2.00 ± 0.50	0.98 ± 0.14	0.15 ± 0.38
	FairHetero (10,0.1,0.1,0.1,0.1,0.1)	1.46 ± 0.32	<b>1.85 ± 0.40</b>	1.03 ± 0.11	<b>0.10 ± 0.32</b>
CIFAR10 Extreme	HeteroFL [9]	1.40 ± 0.48	1.94 ± 0.55	0.71 ± 0.20	0.23 ± 0.48
	q-fll [23]	1.68 ± 0.34	22.00 ± 0.32	0.16 ± 0.18	0.11 ± 0.34
	FedRolex [1]	1.89 ± 0.58	2.53 ± 0.64	0.97 ± 0.34	0.34 ± 0.58
	FjORD [13]	1.56 ± 0.41	2.01 ± 0.44	0.99 ± 0.16	0.16 ± 0.41
	FairHetero (10,0.1,0.1,0.1,0.1,0.1)	1.53 ± 0.36	<b>1.88 ± 0.35</b>	1.03 ± 0.14	<b>0.13 ± 0.36</b>
FEMNIST	HeteroFL [9]	0.75 ± 0.48	1.39 ± 0.64	0.16 ± 0.11	0.24 ± 0.48
	q-fll [23]	0.73 ± 0.33	1.15 ± 0.42	0.31 ± 0.09	0.11 ± 0.33
	FedRolex [1]	0.93 ± 0.52	1.71 ± 0.78	0.33 ± 0.09	0.27 ± 0.52
	FjORD [13]	0.74 ± 0.40	1.28 ± 0.54	0.25 ± 0.09	<b>0.16 ± 0.40</b>
	FairHetero (50,1e-06,1e-05,0.0001,0.001,0.01)	0.60 ± 0.44	<b>1.10 ± 0.50</b>	0.13 ± 0.03	0.20 ± 0.44
SHAKESPEARE	HeteroFL [9]	2.21 ± 0.21	2.47 ± 0.27	1.91 ± 0.08	0.04 ± 0.21
	q-fll [23]	2.30 ± 0.16	2.49 ± 0.19	2.06 ± 0.08	0.02 ± 0.16
	FedRolex [1]	2.28 ± 0.27	2.66 ± 0.37	1.91 ± 0.11	0.07 ± 0.27
	FjORD [13]	2.17 ± 0.16	2.35 ± 0.18	1.97 ± 0.04	<b>0.03 ± 0.16</b>
	FairHetero (0.001,0.1,0.01,0.001,0.0001,1e-05)	2.02 ± 0.17	<b>2.21 ± 0.19</b>	1.81 ± 0.04	0.03 ± 0.17

Bold in Table 3 means maximum.

values. Increasing  $q_m$  reduces the variance within each group but does not address hardware heterogeneity. With increasing  $q$ , there is initially a notable enhancement in the performance of lower architecture groups, but beyond a certain threshold, all groups experience performance degradation, with higher architecture groups being more affected. The optimal  $q$  values for the FEMNIST dataset fall in the range of 10–100. FairHetero has a more pronounced impact on clients with limited resources (e.g., Group 5) compared to those with higher computational power (e.g., Group 1). Overall, FairHetero effectively balances fairness and performance across various computational capabilities for the FEMNIST dataset, addressing heterogeneity in federated learning environments.

Table 4. Performance of FairHetero for MNIST IID Dataset with Different  $q$  and  $q_m$  Values

q	qm1-qm5	Group 1		Group 2		Group 3		Group4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0	0.19 ± 0.12	0.01 ± 0.12	0.18 ± 0.07	0.01 ± 0.07	0.19 ± 0.05	0.00 ± 0.05	0.25 ± 0.02	0.00 ± 0.02	0.58 ± 0.02	0.00 ± 0.02
	0.1,0.1,0.1,0.1,0.1	0.14 ± 0.08	0.01 ± 0.08	0.17 ± 0.06	0.00 ± 0.06	0.20 ± 0.05	0.00 ± 0.05	0.28 ± 0.02	0.00 ± 0.02	0.74 ± 0.02	0.00 ± 0.02
	1,1,1,1,1	0.14 ± 0.06	0.00 ± 0.06	0.22 ± 0.06	0.00 ± 0.06	0.25 ± 0.05	0.00 ± 0.05	0.39 ± 0.02	0.00 ± 0.02	1.11 ± 0.02	0.00 ± 0.02
	10,10,10,10,10	0.26 ± 0.06	0.00 ± 0.06	0.36 ± 0.08	0.01 ± 0.08	0.46 ± 0.05	0.00 ± 0.05	0.85 ± 0.02	0.00 ± 0.02	1.73 ± 0.01	0.00 ± 0.01
	50,50,50,50,50	0.38 ± 0.06	0.00 ± 0.06	0.59 ± 0.06	0.00 ± 0.06	0.94 ± 0.04	0.00 ± 0.04	1.45 ± 0.02	0.00 ± 0.02	2.03 ± 0.01	0.00 ± 0.01
0.1	0,0,0,0	0.16 ± 0.11	0.01 ± 0.11	0.14 ± 0.06	0.00 ± 0.06	0.15 ± 0.06	0.00 ± 0.06	0.24 ± 0.02	0.00 ± 0.02	0.46 ± 0.02	0.00 ± 0.02
	0.1,0.1,0.1,0.1,0.1	0.12 ± 0.07	0.01 ± 0.07	0.14 ± 0.05	0.00 ± 0.05	0.15 ± 0.05	0.00 ± 0.05	0.27 ± 0.02	0.00 ± 0.02	0.58 ± 0.02	0.00 ± 0.02
	1,1,1,1,1	0.14 ± 0.05	0.00 ± 0.05	0.19 ± 0.05	0.00 ± 0.05	0.20 ± 0.04	0.00 ± 0.04	0.41 ± 0.02	0.00 ± 0.02	1.02 ± 0.02	0.00 ± 0.02
	10,10,10,10,10	0.26 ± 0.06	0.00 ± 0.06	0.35 ± 0.08	0.01 ± 0.08	0.44 ± 0.05	0.00 ± 0.05	0.82 ± 0.02	0.00 ± 0.02	1.69 ± 0.01	0.00 ± 0.01
	50,50,50,50,50	0.37 ± 0.06	0.00 ± 0.06	0.58 ± 0.07	0.00 ± 0.07	0.92 ± 0.04	0.00 ± 0.04	1.42 ± 0.02	0.00 ± 0.02	2.01 ± 0.01	0.00 ± 0.01
1	0,0,0,0	0.13 ± 0.08	0.01 ± 0.08	0.15 ± 0.06	0.00 ± 0.06	0.17 ± 0.06	0.00 ± 0.06	0.27 ± 0.03	0.00 ± 0.03	0.40 ± 0.03	0.00 ± 0.03
	0.1,0.1,0.1,0.1,0.1	0.12 ± 0.07	0.00 ± 0.07	0.15 ± 0.05	0.00 ± 0.05	0.17 ± 0.06	0.00 ± 0.06	0.27 ± 0.03	0.00 ± 0.03	0.42 ± 0.03	0.00 ± 0.03
	1,1,1,1,1	0.14 ± 0.06	0.00 ± 0.06	0.20 ± 0.06	0.00 ± 0.06	0.21 ± 0.05	0.00 ± 0.05	0.32 ± 0.03	0.00 ± 0.03	0.58 ± 0.02	0.00 ± 0.02
	10,10,10,10,10	0.27 ± 0.07	0.00 ± 0.07	0.35 ± 0.08	0.01 ± 0.08	0.39 ± 0.05	0.00 ± 0.05	0.59 ± 0.03	0.00 ± 0.03	1.26 ± 0.02	0.00 ± 0.02
	50,50,50,50,50	0.37 ± 0.06	0.00 ± 0.06	0.54 ± 0.07	0.01 ± 0.07	0.73 ± 0.04	0.00 ± 0.04	1.15 ± 0.03	0.00 ± 0.03	1.84 ± 0.01	0.00 ± 0.01
10	0,0,0,0	0.17 ± 0.06	0.00 ± 0.06	0.25 ± 0.07	0.01 ± 0.07	0.26 ± 0.07	0.00 ± 0.07	0.31 ± 0.03	0.00 ± 0.03	0.43 ± 0.04	0.00 ± 0.04
	0.1,0.1,0.1,0.1,0.1	0.17 ± 0.06	0.00 ± 0.06	0.26 ± 0.07	0.01 ± 0.07	0.26 ± 0.07	0.00 ± 0.07	0.31 ± 0.03	0.00 ± 0.03	0.43 ± 0.04	0.00 ± 0.04
	1,1,1,1,1	0.20 ± 0.06	0.00 ± 0.06	0.27 ± 0.08	0.01 ± 0.08	0.28 ± 0.06	0.00 ± 0.06	0.32 ± 0.03	0.00 ± 0.03	0.44 ± 0.04	0.00 ± 0.04
	10,10,10,10,10	0.30 ± 0.07	0.01 ± 0.07	0.35 ± 0.09	0.01 ± 0.09	0.34 ± 0.06	0.00 ± 0.06	0.38 ± 0.03	0.00 ± 0.03	0.52 ± 0.03	0.00 ± 0.03
	50,50,50,50,50	0.37 ± 0.07	0.01 ± 0.07	0.44 ± 0.10	0.01 ± 0.10	0.42 ± 0.07	0.00 ± 0.07	0.47 ± 0.04	0.00 ± 0.04	0.65 ± 0.03	0.00 ± 0.03
50	0,0,0,0	0.35 ± 0.08	0.01 ± 0.07	0.40 ± 0.10	0.01 ± 0.10	0.37 ± 0.07	0.00 ± 0.07	0.39 ± 0.04	0.00 ± 0.04	0.50 ± 0.04	0.00 ± 0.04
	0.1,0.1,0.1,0.1,0.1	0.35 ± 0.08	0.01 ± 0.07	0.40 ± 0.10	0.01 ± 0.10	0.38 ± 0.07	0.00 ± 0.07	0.39 ± 0.04	0.00 ± 0.04	0.50 ± 0.04	0.00 ± 0.04
	1,1,1,1,1	0.35 ± 0.08	0.01 ± 0.07	0.40 ± 0.10	0.01 ± 0.10	0.38 ± 0.07	0.00 ± 0.07	0.39 ± 0.04	0.00 ± 0.04	0.50 ± 0.04	0.00 ± 0.04
	10,10,10,10,10	0.38 ± 0.08	0.01 ± 0.07	0.43 ± 0.11	0.01 ± 0.11	0.39 ± 0.07	0.00 ± 0.07	0.40 ± 0.04	0.00 ± 0.04	0.52 ± 0.03	0.00 ± 0.03
	50,50,50,50,50	0.46 ± 0.09	0.01 ± 0.09	0.49 ± 0.11	0.01 ± 0.11	0.43 ± 0.07	0.00 ± 0.07	0.44 ± 0.04	0.00 ± 0.04	0.59 ± 0.03	0.00 ± 0.03

Table 5. Performance of FairHetero for MNIST Non-IID Dataset with Different  $q$  and  $q_m$  Values

q	qm1-qm5	Group 1		Group 2		Group 3		Group4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0	0.19 ± 0.09	0.01 ± 0.09	0.27 ± 0.12	0.01 ± 0.12	0.26 ± 0.15	0.02 ± 0.15	0.65 ± 0.23	0.05 ± 0.23	1.30 ± 0.34	0.12 ± 0.34
	0.1,0.1,0.1,0.1,0.1	0.23 ± 0.09	0.01 ± 0.09	0.34 ± 0.13	0.02 ± 0.13	0.34 ± 0.17	0.03 ± 0.17	0.93 ± 0.33	0.11 ± 0.33	1.61 ± 0.31	0.10 ± 0.31
	1,1,1,1,1	0.25 ± 0.10	0.01 ± 0.10	0.39 ± 0.13	0.02 ± 0.13	0.42 ± 0.17	0.03 ± 0.17	1.11 ± 0.33	0.11 ± 0.33	1.78 ± 0.27	0.07 ± 0.27
	10,10,10,10,10	0.69 ± 0.10	0.01 ± 0.10	1.01 ± 0.13	0.02 ± 0.13	1.25 ± 0.15	0.02 ± 0.15	1.85 ± 0.22	0.05 ± 0.22	2.17 ± 0.14	0.02 ± 0.14
	50,50,50,50,50	1.51 ± 0.10	0.01 ± 0.10	1.82 ± 0.15	0.02 ± 0.15	2.00 ± 0.08	0.01 ± 0.08	2.17 ± 0.11	0.01 ± 0.11	2.27 ± 0.05	0.00 ± 0.05
0.1	0,0,0,0	0.20 ± 0.10	0.01 ± 0.10	0.28 ± 0.12	0.01 ± 0.12	0.26 ± 0.15	0.02 ± 0.15	0.65 ± 0.23	0.05 ± 0.23	1.28 ± 0.34	0.12 ± 0.34
	0.1,0.1,0.1,0.1,0.1	0.23 ± 0.09	0.01 ± 0.09	0.34 ± 0.13	0.02 ± 0.13	0.33 ± 0.17	0.03 ± 0.17	0.88 ± 0.31	0.09 ± 0.31	1.55 ± 0.32	0.10 ± 0.32
	1,1,1,1,1	0.52 ± 0.08	0.01 ± 0.08	0.69 ± 0.18	0.03 ± 0.18	0.82 ± 0.24	0.06 ± 0.24	1.55 ± 0.35	0.12 ± 0.35	2.05 ± 0.25	0.06 ± 0.25
	10,10,10,10,10	0.69 ± 0.10	0.01 ± 0.10	1.00 ± 0.13	0.02 ± 0.13	1.24 ± 0.15	0.02 ± 0.15	1.84 ± 0.22	0.05 ± 0.22	2.17 ± 0.14	0.02 ± 0.14
	50,50,50,50,50	0.20 ± 0.10	0.01 ± 0.10	0.29 ± 0.13	0.02 ± 0.13	0.26 ± 0.14	0.02 ± 0.14	0.48 ± 0.18	0.03 ± 0.18	0.92 ± 0.3	0.12 ± 0.35
1	0.1,0.1,0.1,0.1,0.1	0.21 ± 0.10	0.01 ± 0.10	0.30 ± 0.13	0.02 ± 0.13	0.27 ± 0.15	0.02 ± 0.15	0.51 ± 0.18	0.03 ± 0.18	0.96 ± 0.35	0.12 ± 0.35
	.0001,.001,.01,.1,1	0.20 ± 0.10	0.01 ± 0.10	0.29 ± 0.12	0.01 ± 0.12	0.28 ± 0.16	0.02 ± 0.16	1.06 ± 0.35	0.12 ± 0.35	1.75 ± 0.29	0.08 ± 0.29
	1,1,1,1,1	0.26 ± 0.10	0.01 ± 0.10	0.37 ± 0.13	0.02 ± 0.13	0.36 ± 0.16	0.02 ± 0.16	0.76 ± 0.21	0.04 ± 0.21	1.29 ± 0.31	0.10 ± 0.31
	0,0,0,0	0.24 ± 0.11	0.01 ± 0.11	0.33 ± 0.14	0.02 ± 0.14	0.30 ± 0.15	0.02 ± 0.15	0.47 ± 0.17	0.03 ± 0.17	0.77 ± 0.31	0.10 ± 0.31
	0.1,0.1,0.1,0.1,0.1	0.24 ± 0.11	0.01 ± 0.11	0.33 ± 0.14	0.02 ± 0.14	0.31 ± 0.15	0.02 ± 0.15	0.48 ± 0.17	0.03 ± 0.17	0.78 ± 0.31	0.10 ± 0.31
10	1,1,1,1,1	0.27 ± 0.12	0.01 ± 0.12	0.38 ± 0.14	0.02 ± 0.14	0.35 ± 0.15	0.02 ± 0.15	0.55 ± 0.16	0.03 ± 0.16	0.84 ± 0.27	0.07 ± 0.27
	.0001,.001,.01,.1,1	0.26 ± 0.11	0.01 ± 0.11	0.37 ± 0.15	0.02 ± 0.15	0.36 ± 0.19	0.04 ± 0.19	1.12 ± 0.35	0.12 ± 0.35	1.61 ± 0.38	0.14 ± 0.38
	10,10,10,10,10	0.58 ± 0.09	0.01 ± 0.09	0.73 ± 0.15	0.02 ± 0.15	0.75 ± 0.14	0.02 ± 0.14	1.08 ± 0.13	0.02 ± 0.13	1.40 ± 0.23	0.05 ± 0.23
	0,0,0,0	0.56 ± 0.15	0.02 ± 0.15	0.68 ± 0.23	0.05 ± 0.23	0.64 ± 0.27	0.07 ± 0.27	0.93 ± 0.28	0.08 ± 0.28	1.18 ± 0.46	0.21 ± 0.46
	0.1,0.1,0.1,0.1,0.1	0.56 ± 0.14	0.02 ± 0.14	0.68 ± 0.23	0.05 ± 0.23	0.65 ± 0.26	0.07 ± 0.26	0.93 ± 0.27	0.07 ± 0.27	1.18 ± 0.45	0.20 ± 0.45
50	1,1,1,1,1	0.67 ± 0.11	0.01 ± 0.11	0.75 ± 0.23	0.05 ± 0.23	0.71 ± 0.24	0.06 ± 0.24	0.98 ± 0.24	0.06 ± 0.24	1.22 ± 0.39	0.15 ± 0.39
	10,10,10,10,10	0.85 ± 0.08	0.01 ± 0.08	1.01 ± 0.18	0.03 ± 0.18	1.03 ± 0.18	0.03 ± 0.18	1.28 ± 0.16	0.03 ± 0.16	1.49 ± 0.25	0.06 ± 0.25
	50,50,50,50,50	1.51 ± 0.11	0.01 ± 0.11	1.65 ± 0.23	0.05 ± 0.23	1.72 ± 0.16	0.03 ± 0.16	1.85 ± 0.16	0.02 ± 0.16	1.99 ± 0.05	0.00 ± 0.05

**Outcome of SHAKESPEARE dataset evaluation.** Table 11 illustrates the average performance of each group in the SHAKESPEARE dataset, displaying the mean and variance of the loss for different  $q$  and  $q_m$  values. Increasing  $q_m$  decreases the variance within each group but does not mitigate hardware heterogeneity. As  $q$  increases, there is an initial enhancement in the performance of lower architecture groups. However, beyond a certain point, all groups experience performance degradation, with higher architecture groups being more affected. The optimal  $q$  values for the SHAKESPEARE dataset range from 0 to 1, indicating a narrow range for RNN models. Overall, FairHetero effectively balances fairness and performance across various computational capabilities for the SHAKESPEARE dataset in RNN models, addressing heterogeneity in federated learning environments.

Table 6. Performance of FairHetero for MNIST Non-IID Extreme Dataset with Different  $q$  and  $q_m$  Values

q	qm1-qm5	Group 1		Group 2		Group 3		Group4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0	0.27 ± 0.13	0.02 ± 0.13	0.25 ± 0.11	0.01 ± 0.11	0.31 ± 0.16	0.03 ± 0.16	0.65 ± 0.22	0.05 ± 0.22	1.29 ± 0.30	0.09 ± 0.30
	0.1,0.1,0.1,0.1,0.1	0.29 ± 0.13	0.02 ± 0.13	0.27 ± 0.11	0.01 ± 0.11	0.33 ± 0.16	0.03 ± 0.16	0.73 ± 0.24	0.06 ± 0.24	1.40 ± 0.27	0.08 ± 0.27
	1,1,1,1,1	0.61 ± 0.10	0.01 ± 0.10	0.66 ± 0.15	0.02 ± 0.15	0.84 ± 0.17	0.03 ± 0.17	1.58 ± 0.31	0.10 ± 0.31	2.03 ± 0.12	0.01 ± 0.12
	10,10,10,10,10	1.27 ± 0.08	0.01 ± 0.08	1.44 ± 0.10	0.01 ± 0.10	1.63 ± 0.12	0.01 ± 0.12	2.03 ± 0.16	0.03 ± 0.16	2.22 ± 0.05	0.00 ± 0.05
	50,50,50,50,50	1.50 ± 0.08	0.01 ± 0.08	1.72 ± 0.08	0.01 ± 0.08	1.90 ± 0.06	0.00 ± 0.06	2.15 ± 0.09	0.01 ± 0.09	2.15 ± 0.09	0.01 ± 0.09
0.1	0,0,0,0,0	0.29 ± 0.13	0.02 ± 0.13	0.26 ± 0.11	0.01 ± 0.11	0.32 ± 0.17	0.03 ± 0.17	0.66 ± 0.22	0.05 ± 0.22	1.28 ± 0.31	0.10 ± 0.31
	0.1,0.1,0.1,0.1,0.1	0.29 ± 0.13	0.02 ± 0.13	0.27 ± 0.11	0.01 ± 0.11	0.33 ± 0.16	0.03 ± 0.16	0.70 ± 0.23	0.05 ± 0.23	1.33 ± 0.29	0.09 ± 0.29
	1,0,1,1,1	0.60 ± 0.10	0.01 ± 0.10	0.65 ± 0.15	0.02 ± 0.15	0.83 ± 0.17	0.03 ± 0.17	1.55 ± 0.31	0.10 ± 0.31	2.01 ± 0.12	0.02 ± 0.12
	10,10,10,10,10	1.74 ± 0.10	0.01 ± 0.10	0.91 ± 0.11	0.01 ± 0.11	1.19 ± 0.11	0.01 ± 0.11	1.81 ± 0.20	0.04 ± 0.20	2.13 ± 0.08	0.01 ± 0.08
	50,50,50,50,50	1.50 ± 0.08	0.01 ± 0.08	1.72 ± 0.08	0.01 ± 0.08	1.90 ± 0.06	0.00 ± 0.06	2.15 ± 0.09	0.01 ± 0.09	2.25 ± 0.04	0.00 ± 0.04
1	0,0,0,0,0	0.29 ± 0.13	0.02 ± 0.13	0.27 ± 0.11	0.01 ± 0.11	0.31 ± 0.17	0.03 ± 0.17	0.50 ± 0.16	0.03 ± 0.16	0.89 ± 0.38	0.15 ± 0.38
	0.1,0.1,0.1,0.1,0.1	0.31 ± 0.13	0.02 ± 0.13	0.28 ± 0.11	0.01 ± 0.11	0.32 ± 0.17	0.03 ± 0.17	0.53 ± 0.17	0.03 ± 0.17	0.94 ± 0.38	0.14 ± 0.38
	1,0,1,1,1	0.56 ± 0.11	0.01 ± 0.11	0.57 ± 0.16	0.02 ± 0.16	0.69 ± 0.15	0.02 ± 0.15	1.27 ± 0.32	0.10 ± 0.32	1.77 ± 0.18	0.03 ± 0.18
	10,10,10,10,10	0.72 ± 0.10	0.01 ± 0.10	0.85 ± 0.11	0.01 ± 0.11	1.10 ± 0.10	0.01 ± 0.10	1.68 ± 0.20	0.04 ± 0.20	2.05 ± 0.09	0.01 ± 0.09
	50,50,50,50,50	1.49 ± 0.08	0.01 ± 0.08	1.70 ± 0.08	0.01 ± 0.08	1.88 ± 0.06	0.00 ± 0.06	2.13 ± 0.09	0.01 ± 0.09	2.25 ± 0.04	0.00 ± 0.04
10	0,0,0,0,0	0.35 ± 0.14	0.02 ± 0.14	0.32 ± 0.13	0.02 ± 0.13	0.35 ± 0.18	0.03 ± 0.18	0.49 ± 0.17	0.03 ± 0.17	0.78 ± 0.41	0.17 ± 0.41
	0.001,0.01,0.1,0.1,10	0.36 ± 0.14	0.02 ± 0.14	0.36 ± 0.15	0.02 ± 0.15	0.43 ± 0.18	0.03 ± 0.18	1.14 ± 0.37	0.14 ± 0.37	1.65 ± 0.28	0.08 ± 0.28
	0.1,0.1,0.1,0.1,0.1	0.36 ± 0.14	0.02 ± 0.14	0.33 ± 0.13	0.02 ± 0.13	0.35 ± 0.17	0.03 ± 0.17	0.50 ± 0.17	0.03 ± 0.17	0.79 ± 0.40	0.16 ± 0.40
	1,1,1,1,1	0.41 ± 0.13	0.02 ± 0.13	0.38 ± 0.13	0.02 ± 0.13	0.40 ± 0.16	0.03 ± 0.16	0.56 ± 0.16	0.02 ± 0.16	0.86 ± 0.33	0.11 ± 0.33
	10,10,10,10,10	0.64 ± 0.11	0.01 ± 0.11	0.66 ± 0.13	0.02 ± 0.13	0.76 ± 0.10	0.01 ± 0.10	1.04 ± 0.17	0.03 ± 0.17	1.45 ± 0.24	0.06 ± 0.24
50	0,0,0,0,0	0.61 ± 0.21	0.05 ± 0.21	0.59 ± 0.24	0.06 ± 0.24	0.65 ± 0.22	0.05 ± 0.22	0.92 ± 0.31	0.10 ± 0.31	1.19 ± 0.48	0.23 ± 0.48
	0.1,0.1,0.1,0.1,0.1	0.61 ± 0.21	0.04 ± 0.21	0.59 ± 0.24	0.06 ± 0.24	0.65 ± 0.22	0.05 ± 0.22	0.92 ± 0.31	0.10 ± 0.31	1.20 ± 0.47	0.22 ± 0.47
	1,1,1,1,1	0.63 ± 0.20	0.04 ± 0.20	0.60 ± 0.21	0.04 ± 0.21	0.68 ± 0.19	0.04 ± 0.19	0.93 ± 0.28	0.08 ± 0.28	1.22 ± 0.41	0.17 ± 0.41
	10,10,10,10,10	0.84 ± 0.15	0.02 ± 0.15	0.85 ± 0.14	0.02 ± 0.14	0.98 ± 0.11	0.01 ± 0.11	1.26 ± 0.22	0.05 ± 0.22	1.54 ± 0.26	0.07 ± 0.26

Table 7. Performance of FairHetero for CIFAR10 IID Dataset with Different  $q$  and  $q_m$  Values

q	qm1-qm5	Group 1		Group 2		Group 3		Group4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0,0	1.18 ± 0.14	0.02 ± 0.14	1.21 ± 0.11	0.01 ± 0.11	1.37 ± 0.08	0.01 ± 0.08	1.85 ± 0.02	0.00 ± 0.02	2.03 ± 0.02	0.00 ± 0.02
	0.1,0.1,0.1,0.1,0.1	1.17 ± 0.13	0.02 ± 0.13	1.22 ± 0.11	0.01 ± 0.11	1.38 ± 0.08	0.01 ± 0.08	1.86 ± 0.02	0.00 ± 0.02	2.03 ± 0.02	0.00 ± 0.02
	1,1,1,1,1	1.16 ± 0.12	0.01 ± 0.12	1.27 ± 0.10	0.01 ± 0.10	1.44 ± 0.07	0.00 ± 0.07	1.93 ± 0.02	0.00 ± 0.02	2.08 ± 0.02	0.00 ± 0.02
	10,10,10,10,10	1.40 ± 0.10	0.01 ± 0.10	1.50 ± 0.09	0.01 ± 0.09	1.67 ± 0.04	0.00 ± 0.04	2.18 ± 0.01	0.00 ± 0.01	2.24 ± 0.01	0.00 ± 0.01
	50,50,50,50,50	1.68 ± 0.08	0.01 ± 0.08	1.79 ± 0.05	0.00 ± 0.05	1.98 ± 0.04	0.00 ± 0.04	2.28 ± 0.00	0.00 ± 0.00	2.29 ± 0.00	0.00 ± 0.00
0.1	0,0,0,0,0	1.18 ± 0.14	0.02 ± 0.14	1.21 ± 0.11	0.01 ± 0.11	1.37 ± 0.08	0.01 ± 0.08	1.84 ± 0.02	0.00 ± 0.02	2.02 ± 0.02	0.00 ± 0.02
	0.1,0.1,0.1,0.1,0.1	1.17 ± 0.13	0.02 ± 0.13	1.22 ± 0.11	0.01 ± 0.11	1.37 ± 0.08	0.01 ± 0.08	1.85 ± 0.02	0.00 ± 0.02	2.02 ± 0.02	0.00 ± 0.02
	1,1,1,1,1	1.16 ± 0.11	0.01 ± 0.11	1.27 ± 0.10	0.01 ± 0.10	1.43 ± 0.07	0.00 ± 0.07	1.92 ± 0.02	0.00 ± 0.02	2.07 ± 0.02	0.00 ± 0.02
	10,10,10,10,10	1.40 ± 0.10	0.01 ± 0.10	1.50 ± 0.09	0.01 ± 0.09	1.67 ± 0.04	0.00 ± 0.04	2.18 ± 0.01	0.00 ± 0.01	2.24 ± 0.01	0.00 ± 0.01
	50,50,50,50,50	1.68 ± 0.08	0.01 ± 0.08	1.79 ± 0.05	0.00 ± 0.05	1.98 ± 0.04	0.00 ± 0.04	2.28 ± 0.00	0.00 ± 0.00	2.29 ± 0.00	0.00 ± 0.00
1	0,0,0,0,0	1.16 ± 0.13	0.02 ± 0.13	1.22 ± 0.11	0.01 ± 0.11	1.36 ± 0.08	0.01 ± 0.08	1.80 ± 0.02	0.00 ± 0.02	1.97 ± 0.02	0.00 ± 0.02
	.01,.01,.01,.01,.01	1.16 ± 0.13	0.02 ± 0.13	1.21 ± 0.11	0.01 ± 0.11	1.36 ± 0.08	0.01 ± 0.08	1.80 ± 0.02	0.00 ± 0.02	1.97 ± 0.02	0.00 ± 0.02
	1,1,1,1,1	1.17 ± 0.11	0.01 ± 0.11	1.26 ± 0.10	0.01 ± 0.10	1.42 ± 0.07	0.01 ± 0.07	1.86 ± 0.02	0.00 ± 0.02	2.02 ± 0.02	0.00 ± 0.02
	10,10,10,10,10	1.40 ± 0.10	0.01 ± 0.10	1.50 ± 0.09	0.01 ± 0.09	1.66 ± 0.04	0.01 ± 0.04	2.14 ± 0.01	0.00 ± 0.01	2.21 ± 0.01	0.00 ± 0.01
	50,50,50,50,50	1.68 ± 0.08	0.01 ± 0.08	1.79 ± 0.05	0.00 ± 0.05	1.98 ± 0.04	0.00 ± 0.04	2.28 ± 0.00	0.00 ± 0.00	2.29 ± 0.00	0.00 ± 0.00
10	0,0,0,0,0	1.19 ± 0.11	0.01 ± 0.11	1.25 ± 0.10	0.01 ± 0.10	1.39 ± 0.08	0.01 ± 0.08	1.74 ± 0.03	0.00 ± 0.03	1.87 ± 0.02	0.00 ± 0.02
	0.1,0.1,0.1,0.1,0.1	1.19 ± 0.11	0.01 ± 0.11	1.26 ± 0.10	0.01 ± 0.10	1.39 ± 0.08	0.01 ± 0.08	1.74 ± 0.03	0.00 ± 0.03	1.87 ± 0.02	0.00 ± 0.02
	1,1,1,1,1	1.23 ± 0.11	0.01 ± 0.11	1.30 ± 0.09	0.01 ± 0.09	1.43 ± 0.08	0.01 ± 0.08	1.76 ± 0.03	0.00 ± 0.03	1.88 ± 0.02	0.00 ± 0.02
	10,10,10,10,10	1.45 ± 0.10	0.01 ± 0.10	1.51 ± 0.08	0.01 ± 0.08	1.62 ± 0.05	0.00 ± 0.05	1.91 ± 0.02	0.00 ± 0.02	2.02 ± 0.02	0.00 ± 0.02
	50,50,50,50,50	1.77 ± 0.07	0.00 ± 0.07	1.80 ± 0.05	0.00 ± 0.05	1.85 ± 0.04	0.00 ± 0.04	2.01 ± 0.02	0.00 ± 0.02	2.06 ± 0.02	0.00 ± 0.02

**Key takeaways.** The values of  $q$  and  $q_m$  are not overly sensitive regarding performance, generally exhibiting a wide range where they ensure both fairness and performance. This range was found to be broader for MLP and CNN model architectures, as seen in the MNIST, CIFAR10, and FEMNIST datasets, while narrower for the RNN model, as observed in the SHAKESPEARE dataset. In most cases, even a small value of  $q$  and  $q_m$  can ensure improved fairness with added performance.

## 7 Conclusion

In this article, we proposed a novel FL method, FairHetero, that promotes fairness among clients with heterogeneous hardware or model architectures, ensuring balance and equity in model training. Our approach offers tunable fairness, addressing both data and hardware heterogeneity. We conducted an extensive theoretical and experimental evaluation and demonstrated that FairHetero can reduce performance variability among participating devices.

Table 8. Performance of FairHetero for CIFAR10 Non-IID Dataset with Different  $q$  and  $q_m$  Values

q	qm1-qm5	Group 1		Group 2		Group 3		Group4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0	1.14 ± 0.28	0.08 ± 0.28	1.29 ± 0.29	0.09 ± 0.29	1.52 ± 0.30	0.09 ± 0.30	1.69 ± 0.26	0.07 ± 0.26	1.88 ± 0.23	0.05 ± 0.23
	0.1,0.1,0.1,0.1,0.1	1.16 ± 0.27	0.07 ± 0.27	1.32 ± 0.28	0.08 ± 0.28	1.54 ± 0.30	0.09 ± 0.30	1.72 ± 0.25	0.06 ± 0.25	1.90 ± 0.23	0.05 ± 0.23
	1,1,1,1,1	1.31 ± 0.23	0.05 ± 0.23	1.46 ± 0.25	0.06 ± 0.25	1.66 ± 0.29	0.09 ± 0.29	1.86 ± 0.23	0.05 ± 0.23	2.01 ± 0.19	0.03 ± 0.19
	10,10,10,10,10	1.67 ± 0.10	0.01 ± 0.10	1.78 ± 0.16	0.02 ± 0.16	1.95 ± 0.19	0.04 ± 0.19	2.14 ± 0.16	0.03 ± 0.16	2.22 ± 0.09	0.01 ± 0.09
	50,50,50,50,50	1.94 ± 0.12	0.02 ± 0.12	2.09 ± 0.18	0.03 ± 0.18	2.20 ± 0.13	0.02 ± 0.13	2.24 ± 0.10	0.01 ± 0.10	2.28 ± 0.06	0.00 ± 0.06
0.1	0,0,0,0	1.14 ± 0.28	0.08 ± 0.28	1.29 ± 0.29	0.09 ± 0.29	1.51 ± 0.30	0.09 ± 0.30	1.69 ± 0.26	0.07 ± 0.26	1.87 ± 0.23	0.05 ± 0.23
	0.1,0.1,0.1,0.1,0.1	1.16 ± 0.27	0.07 ± 0.27	1.32 ± 0.28	0.08 ± 0.28	1.54 ± 0.30	0.09 ± 0.30	1.71 ± 0.25	0.06 ± 0.25	1.90 ± 0.23	0.05 ± 0.23
	1,1,1,1,1	1.31 ± 0.23	0.05 ± 0.23	1.46 ± 0.25	0.06 ± 0.25	1.66 ± 0.29	0.09 ± 0.29	1.85 ± 0.23	0.05 ± 0.23	2.01 ± 0.19	0.03 ± 0.19
	10,10,10,10,10	1.67 ± 0.10	0.01 ± 0.10	1.78 ± 0.16	0.02 ± 0.16	1.95 ± 0.19	0.04 ± 0.19	2.14 ± 0.16	0.03 ± 0.16	2.22 ± 0.09	0.01 ± 0.09
	50,50,50,50,50	1.94 ± 0.12	0.02 ± 0.12	2.09 ± 0.18	0.03 ± 0.18	2.20 ± 0.13	0.02 ± 0.13	2.24 ± 0.10	0.01 ± 0.10	2.28 ± 0.06	0.00 ± 0.06
1	0,0,0,0	1.17 ± 0.26	0.07 ± 0.26	1.32 ± 0.29	0.08 ± 0.29	1.52 ± 0.30	0.09 ± 0.30	1.67 ± 0.25	0.06 ± 0.25	1.85 ± 0.22	0.05 ± 0.22
	0.1,0.1,0.1,0.1,0.1	1.17 ± 0.26	0.07 ± 0.26	1.32 ± 0.29	0.08 ± 0.29	1.52 ± 0.30	0.09 ± 0.30	1.67 ± 0.25	0.06 ± 0.25	1.85 ± 0.22	0.05 ± 0.22
	1,1,1,1,1	1.32 ± 0.23	0.05 ± 0.23	1.46 ± 0.26	0.07 ± 0.26	1.64 ± 0.29	0.09 ± 0.29	1.81 ± 0.23	0.05 ± 0.23	1.96 ± 0.19	0.04 ± 0.19
	10,10,10,10,10	1.68 ± 0.10	0.01 ± 0.10	1.77 ± 0.16	0.03 ± 0.16	1.93 ± 0.20	0.04 ± 0.20	2.12 ± 0.17	0.03 ± 0.17	2.20 ± 0.09	0.01 ± 0.09
	50,50,50,50,50	2.03 ± 0.08	0.01 ± 0.08	2.07 ± 0.16	0.03 ± 0.16	2.06 ± 0.15	0.02 ± 0.15	2.12 ± 0.15	0.02 ± 0.15	2.16 ± 0.11	0.01 ± 0.11
10	0,0,0,0	1.23 ± 0.26	0.07 ± 0.26	1.30 ± 0.29	0.08 ± 0.29	1.48 ± 0.29	0.08 ± 0.29	1.57 ± 0.24	0.06 ± 0.24	1.71 ± 0.23	0.05 ± 0.23
	0.1,0.1,0.1,0.1,0.1	1.25 ± 0.26	0.07 ± 0.26	1.33 ± 0.30	0.09 ± 0.30	1.49 ± 0.29	0.08 ± 0.29	1.58 ± 0.24	0.06 ± 0.24	1.72 ± 0.23	0.05 ± 0.23
	1,1,1,1,1	1.37 ± 0.23	0.05 ± 0.23	1.45 ± 0.26	0.07 ± 0.26	1.57 ± 0.26	0.07 ± 0.26	1.65 ± 0.23	0.05 ± 0.23	1.78 ± 0.19	0.03 ± 0.19
	10,10,10,10,10	1.73 ± 0.09	0.01 ± 0.09	1.78 ± 0.16	0.03 ± 0.16	1.87 ± 0.18	0.03 ± 0.18	1.96 ± 0.16	0.02 ± 0.16	2.05 ± 0.10	0.01 ± 0.10
	50,50,50,50,50	2.03 ± 0.08	0.01 ± 0.08	2.07 ± 0.16	0.03 ± 0.16	2.06 ± 0.15	0.02 ± 0.15	2.12 ± 0.15	0.02 ± 0.15	2.16 ± 0.11	0.01 ± 0.11
50	0,0,0,0	1.37 ± 0.25	0.06 ± 0.25	1.38 ± 0.29	0.08 ± 0.29	1.50 ± 0.29	0.09 ± 0.29	1.56 ± 0.25	0.06 ± 0.25	1.66 ± 0.24	0.06 ± 0.24
	0.1,0.1,0.1,0.1,0.1	1.46 ± 0.22	0.05 ± 0.22	1.47 ± 0.26	0.07 ± 0.26	1.57 ± 0.27	0.07 ± 0.27	1.61 ± 0.23	0.05 ± 0.23	1.71 ± 0.20	0.04 ± 0.20
	1,1,1,1,1	1.47 ± 0.22	0.05 ± 0.22	1.47 ± 0.26	0.07 ± 0.26	1.57 ± 0.27	0.07 ± 0.27	1.61 ± 0.23	0.05 ± 0.23	1.71 ± 0.20	0.04 ± 0.20
	10,10,10,10,10	1.77 ± 0.08	0.01 ± 0.08	1.77 ± 0.15	0.02 ± 0.15	1.84 ± 0.17	0.03 ± 0.17	1.88 ± 0.14	0.02 ± 0.14	1.95 ± 0.10	0.01 ± 0.10
	50,50,50,50,50	2.03 ± 0.08	0.01 ± 0.08	2.07 ± 0.16	0.03 ± 0.16	2.06 ± 0.15	0.02 ± 0.15	2.12 ± 0.15	0.02 ± 0.15	2.16 ± 0.11	0.01 ± 0.11

Table 9. Performance of FairHetero for CIFAR10 Non-IID Extreme Dataset with Different  $q$  and  $q_m$  Values

q	qm1-qm5	Group 1		Group 2		Group 3		Group4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0	1.29 ± 0.34	0.12 ± 0.34	1.32 ± 0.38	0.14 ± 0.38	1.56 ± 0.44	0.19 ± 0.44	1.77 ± 0.22	0.05 ± 0.22	1.88 ± 0.23	0.05 ± 0.23
	0.1,0.1,0.1,0.1,0.1	1.32 ± 0.33	0.11 ± 0.33	1.36 ± 0.37	0.14 ± 0.37	1.58 ± 0.43	0.18 ± 0.43	1.79 ± 0.21	0.04 ± 0.21	1.89 ± 0.22	0.05 ± 0.22
	1,1,1,1,1	1.45 ± 0.29	0.08 ± 0.29	1.49 ± 0.33	0.11 ± 0.33	1.66 ± 0.36	0.13 ± 0.36	1.90 ± 0.18	0.03 ± 0.18	2.00 ± 0.18	0.03 ± 0.18
	10,10,10,10,10	1.75 ± 0.26	0.07 ± 0.26	1.80 ± 0.21	0.04 ± 0.21	1.93 ± 0.21	0.04 ± 0.21	2.16 ± 0.10	0.01 ± 0.10	2.21 ± 0.09	0.01 ± 0.09
	50,50,50,50,50	1.96 ± 0.25	0.06 ± 0.25	2.12 ± 0.17	0.03 ± 0.17	2.14 ± 0.14	0.02 ± 0.14	2.25 ± 0.06	0.00 ± 0.06	2.27 ± 0.04	0.00 ± 0.04
0.1	0,0,0,0	1.29 ± 0.34	0.12 ± 0.34	1.32 ± 0.38	0.14 ± 0.38	1.55 ± 0.44	0.19 ± 0.44	1.76 ± 0.22	0.05 ± 0.22	1.87 ± 0.23	0.05 ± 0.23
	0.1,0.1,0.1,0.1,0.1	1.32 ± 0.33	0.11 ± 0.33	1.36 ± 0.38	0.14 ± 0.38	1.57 ± 0.43	0.18 ± 0.43	1.78 ± 0.21	0.04 ± 0.21	1.89 ± 0.22	0.05 ± 0.22
	1,1,1,1,1	1.46 ± 0.29	0.08 ± 0.29	1.49 ± 0.33	0.11 ± 0.33	1.66 ± 0.36	0.13 ± 0.36	1.90 ± 0.18	0.03 ± 0.18	1.99 ± 0.18	0.03 ± 0.18
	10,10,10,10,10	1.76 ± 0.26	0.07 ± 0.26	1.80 ± 0.21	0.05 ± 0.21	1.92 ± 0.21	0.04 ± 0.21	2.16 ± 0.10	0.01 ± 0.10	2.20 ± 0.09	0.01 ± 0.09
	50,50,50,50,50	1.96 ± 0.25	0.06 ± 0.25	2.12 ± 0.17	0.03 ± 0.17	2.14 ± 0.14	0.02 ± 0.14	2.25 ± 0.06	0.00 ± 0.06	2.27 ± 0.04	0.00 ± 0.04
1	0,0,0,0	1.30 ± 0.34	0.11 ± 0.34	1.33 ± 0.38	0.14 ± 0.38	1.54 ± 0.43	0.19 ± 0.43	1.73 ± 0.22	0.05 ± 0.22	1.82 ± 0.24	0.06 ± 0.24
	0.1,0.1,0.1,0.1,0.1	1.33 ± 0.33	0.11 ± 0.33	1.37 ± 0.37	0.14 ± 0.37	1.56 ± 0.42	0.18 ± 0.42	1.75 ± 0.22	0.05 ± 0.22	1.84 ± 0.23	0.05 ± 0.23
	1,1,1,1,1	1.47 ± 0.29	0.08 ± 0.29	1.50 ± 0.33	0.11 ± 0.33	1.65 ± 0.36	0.13 ± 0.36	1.86 ± 0.18	0.03 ± 0.18	1.93 ± 0.19	0.04 ± 0.19
	10,10,10,10,10	1.76 ± 0.26	0.07 ± 0.26	1.80 ± 0.21	0.05 ± 0.21	1.92 ± 0.21	0.04 ± 0.21	2.16 ± 0.10	0.01 ± 0.10	2.20 ± 0.09	0.01 ± 0.09
	50,50,50,50,50	1.96 ± 0.25	0.06 ± 0.25	2.12 ± 0.17	0.03 ± 0.17	2.14 ± 0.14	0.02 ± 0.14	2.25 ± 0.06	0.00 ± 0.06	2.27 ± 0.04	0.00 ± 0.04
10	0,0,0,0	1.36 ± 0.32	0.10 ± 0.32	1.38 ± 0.38	0.15 ± 0.38	1.51 ± 0.43	0.19 ± 0.43	1.64 ± 0.23	0.05 ± 0.23	1.70 ± 0.29	0.08 ± 0.29
	0.1,0.1,0.1,0.1,0.1	1.38 ± 0.31	0.10 ± 0.31	1.40 ± 0.38	0.14 ± 0.38	1.52 ± 0.42	0.18 ± 0.42	1.65 ± 0.23	0.05 ± 0.23	1.71 ± 0.28	0.08 ± 0.28
	1,1,1,1,1	1.50 ± 0.27	0.07 ± 0.27	1.51 ± 0.33	0.11 ± 0.33	1.58 ± 0.34	0.12 ± 0.34	1.72 ± 0.20	0.04 ± 0.20	1.75 ± 0.24	0.06 ± 0.24
	10,10,10,10,10	1.80 ± 0.21	0.04 ± 0.21	1.78 ± 0.21	0.04 ± 0.21	1.82 ± 0.19	0.04 ± 0.19	1.98 ± 0.14	0.02 ± 0.14	1.97 ± 0.12	0.01 ± 0.12
	50,50,50,50,50	1.46 ± 0.30	0.09 ± 0.30	1.45 ± 0.38	0.15 ± 0.38	1.52 ± 0.43	0.18 ± 0.43	1.64 ± 0.24	0.06 ± 0.24	1.66 ± 0.31	0.09 ± 0.31
50	0,0,0,0	1.56 ± 0.26	0.07 ± 0.26	1.52 ± 0.33	0.11 ± 0.33	1.57 ± 0.35	0.13 ± 0.35	1.68 ± 0.23	0.05 ± 0.23	1.69 ± 0.27	0.07 ± 0.27
	0.1,0.1,0.1,0.1,0.1	1.83 ± 0.23	0.05 ± 0.23	1.77 ± 0.22	0.05 ± 0.22	1.78 ± 0.18	0.03 ± 0.18	1.90 ± 0.17	0.03 ± 0.17	1.86 ± 0.15	0.02 ± 0.15
	1,1,1,1,1	1.56 ± 0.26	0.07 ± 0.26	1.52 ± 0.33	0.11 ± 0.33	1.57 ± 0.35	0.13 ± 0.35	1.68 ± 0.23	0.05 ± 0.23	1.69 ± 0.27	0.07 ± 0.27
	10,10,10,10,10	1.83 ± 0.23	0.05 ± 0.23	1.77 ± 0.22	0.05 ± 0.22	1.78 ± 0.18	0.03 ± 0.18	1.90 ± 0.17	0.03 ± 0.17	1.86 ± 0.15	0.02 ± 0.15
	50,50,50,50,50	2.04 ± 0.21	0.04 ± 0.21	2.04 ± 0.14	0.02 ± 0.14	2.01 ± 0.17	0.03 ± 0.17	2.12 ± 0.13	0.02 ± 0.13	2.10 ± 0.10	0.01 ± 0.10

**Future directions.** This article opens up several future research directions toward achieving FL fairness under a heterogeneous architecture setting. *First*, as identified in Section 4.1, FairHetero requires hyperparameter tuning that adds to FL overhead. However, given that our design goal is to reduce variance in loss, we can utilize the distribution of clients' training loss to tune the parameters  $q$  and  $q_m$ . Albeit, such an approach will require clients to report their training losses along with model updates to the aggregation server. *Second*, different approaches have been used to address heterogeneous architectures in FL, with the end goal of improving overall performance. Adapting a fairness tradeoff mechanism such as FairHetero, in addition to handling heterogeneous architecture, requires further investigation. *Third*, even though FairHetero introduces the notion of hardware fairness, the fairness goal is still driven by model performance, yet there are other important aspects of fairness such as the environmental impact of carbon emission and water consumption of FL clients' local training. Hence, future works addressing fairness will benefit from taking a holistic fairness approach for the federation.

Table 10. Performance of FairHetero for FEMNIST Dataset with Different  $q$  and  $q_m$  Values

$q$	qm1-qm5	Group 1		Group 2		Group 3		Group 4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0	0.45 ± 0.45	0.20 ± 0.45	0.45 ± 0.42	0.17 ± 0.42	0.56 ± 0.32	0.10 ± 0.32	1.04 ± 0.30	0.09 ± 0.30	1.25 ± 0.24	0.06 ± 0.24
	0.1,0.1,0.1,0.1,0.1	0.45 ± 0.45	0.20 ± 0.45	0.45 ± 0.42	0.17 ± 0.42	0.56 ± 0.32	0.10 ± 0.32	1.04 ± 0.30	0.09 ± 0.30	1.25 ± 0.24	0.06 ± 0.24
	1,1,1,1,1	0.45 ± 0.44	0.19 ± 0.44	0.45 ± 0.41	0.17 ± 0.41	0.57 ± 0.31	0.10 ± 0.31	1.04 ± 0.30	0.09 ± 0.30	1.25 ± 0.24	0.06 ± 0.24
	10,10,10,10,10	0.51 ± 0.41	0.17 ± 0.41	0.51 ± 0.40	0.16 ± 0.40	0.63 ± 0.30	0.09 ± 0.30	1.10 ± 0.28	0.08 ± 0.28	1.30 ± 0.23	0.05 ± 0.23
0.1	50,50,50,50,50	0.46 ± 0.09	0.01 ± 0.09	0.49 ± 0.11	0.01 ± 0.11	0.43 ± 0.07	0.01 ± 0.07	0.44 ± 0.04	0.00 ± 0.04	0.59 ± 0.03	0.00 ± 0.03
	0,0,0,0,0	0.45 ± 0.45	0.20 ± 0.45	0.45 ± 0.42	0.17 ± 0.42	0.56 ± 0.32	0.10 ± 0.32	1.04 ± 0.30	0.09 ± 0.30	1.25 ± 0.24	0.06 ± 0.24
	0.1,0.1,0.1,0.1,0.1	0.45 ± 0.45	0.20 ± 0.45	0.45 ± 0.42	0.17 ± 0.42	0.56 ± 0.32	0.10 ± 0.32	1.03 ± 0.31	0.09 ± 0.31	1.24 ± 0.24	0.06 ± 0.24
	.0001,0.001,0.01,0.1,1	0.45 ± 0.45	0.20 ± 0.45	0.45 ± 0.42	0.17 ± 0.42	0.56 ± 0.32	0.10 ± 0.32	1.04 ± 0.31	0.09 ± 0.31	1.24 ± 0.24	0.06 ± 0.24
1	0,0,0,0,0	0.45 ± 0.46	0.21 ± 0.46	0.45 ± 0.43	0.18 ± 0.43	0.55 ± 0.33	0.11 ± 0.33	0.98 ± 0.32	0.10 ± 0.32	1.17 ± 0.26	0.07 ± 0.26
	1,0.1,0.1,0.1,0.1	0.46 ± 0.45	0.21 ± 0.45	0.45 ± 0.43	0.18 ± 0.43	0.55 ± 0.33	0.11 ± 0.33	0.98 ± 0.32	0.10 ± 0.32	1.18 ± 0.26	0.07 ± 0.26
	1,1,0.1,0.01,0.001	0.46 ± 0.45	0.21 ± 0.46	0.45 ± 0.43	0.18 ± 0.43	0.55 ± 0.33	0.11 ± 0.33	0.98 ± 0.32	0.10 ± 0.32	1.18 ± 0.26	0.07 ± 0.26
	.001,0.1,1,1,10	0.46 ± 0.46	0.21 ± 0.45	0.45 ± 0.43	0.18 ± 0.43	0.55 ± 0.33	0.11 ± 0.33	0.98 ± 0.31	0.10 ± 0.31	1.19 ± 0.25	0.06 ± 0.25
10	0,0,0,0,0	0.48 ± 0.51	0.26 ± 0.51	0.46 ± 0.47	0.22 ± 0.47	0.55 ± 0.37	0.14 ± 0.37	0.88 ± 0.37	0.14 ± 0.37	1.04 ± 0.31	0.10 ± 0.31
	0.1,0.1,0.1,0.1,0.1	0.48 ± 0.51	0.26 ± 0.51	0.47 ± 0.47	0.22 ± 0.47	0.55 ± 0.37	0.14 ± 0.37	0.88 ± 0.37	0.14 ± 0.37	1.04 ± 0.31	0.10 ± 0.31
	.0001,0.001,0.01,1,1	0.48 ± 0.51	0.26 ± 0.51	0.47 ± 0.47	0.22 ± 0.47	0.55 ± 0.37	0.14 ± 0.37	0.88 ± 0.37	0.14 ± 0.37	1.04 ± 0.31	0.10 ± 0.31
	10,1,1,1,1	0.52 ± 0.43	0.19 ± 0.43	0.52 ± 0.42	0.17 ± 0.42	0.64 ± 0.34	0.11 ± 0.34	0.91 ± 0.35	0.12 ± 0.35	1.07 ± 0.30	0.09 ± 0.30
50	10,10,10,10,10	0.51 ± 0.43	0.19 ± 0.43	0.50 ± 0.42	0.18 ± 0.42	0.62 ± 0.32	0.10 ± 0.32	0.94 ± 0.32	0.10 ± 0.32	1.10 ± 0.28	0.08 ± 0.28
	0,0,0,0,0	0.49 ± 0.52	0.27 ± 0.52	0.48 ± 0.48	0.23 ± 0.48	0.55 ± 0.38	0.15 ± 0.38	0.71 ± 0.41	0.16 ± 0.41	0.78 ± 0.31	0.10 ± 0.31
	.001,0.1,1,1,10	0.49 ± 0.51	0.26 ± 0.51	0.48 ± 0.47	0.22 ± 0.47	0.55 ± 0.37	0.14 ± 0.37	0.74 ± 0.38	0.15 ± 0.38	0.83 ± 0.28	.08 ± 0.28
	.001,1,1,10,50	0.50 ± 0.49	0.24 ± 0.49	0.49 ± 0.46	0.21 ± 0.46	0.57 ± 0.36	0.13 ± 0.36	0.83 ± 0.35	0.13 ± 0.35	0.95 ± 0.26	0.07 ± 0.26
	50,50,50,50,50	0.61 ± 0.40	0.16 ± 0.40	0.63 ± 0.39	0.15 ± 0.39	0.70 ± 0.30	0.09 ± 0.30	0.91 ± 0.33	0.11 ± 0.33	1.00 ± 0.24	0.06 ± 0.24

Table 11. Performance of FairHetero for SHAKESPEARE Dataset with Different  $q$  and  $q_m$  Values

$q$	qm1-qm5	Group 1		Group 2		Group 3		Group 4		Group 5	
		Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
0	0,0,0,0,0	1.82 ± 0.11	0.01 ± 0.11	1.91 ± 0.07	0.00 ± 0.07	1.98 ± 0.08	0.01 ± 0.08	2.14 ± 0.06	0.00 ± 0.06	2.23 ± 0.08	0.01 ± 0.08
	0.1,0.1,0.1,0.1,0.1	1.82 ± 0.11	0.01 ± 0.11	1.92 ± 0.07	0.00 ± 0.07	2.00 ± 0.08	0.01 ± 0.08	2.15 ± 0.06	0.00 ± 0.06	2.24 ± 0.08	0.01 ± 0.08
	1,1,1,1,1	1.92 ± 0.12	0.01 ± 0.12	2.03 ± 0.07	0.00 ± 0.07	2.11 ± 0.08	0.01 ± 0.08	2.24 ± 0.06	0.00 ± 0.06	2.32 ± 0.08	0.01 ± 0.08
0.001	0,0,0,0,0	1.82 ± 0.11	0.01 ± 0.11	1.91 ± 0.07	0.00 ± 0.07	1.98 ± 0.08	0.01 ± 0.08	2.14 ± 0.06	0.00 ± 0.06	2.23 ± 0.08	0.01 ± 0.08
	.001,0.001,0.001,0.001,0.001	1.82 ± 0.11	0.01 ± 0.11	1.91 ± 0.07	0.00 ± 0.07	1.98 ± 0.08	0.01 ± 0.08	2.14 ± 0.06	0.00 ± 0.06	2.23 ± 0.08	0.01 ± 0.08
	0.1,0.1,0.1,0.1,0.1	1.82 ± 0.11	0.01 ± 0.11	1.91 ± 0.07	0.00 ± 0.07	1.98 ± 0.08	0.01 ± 0.08	2.14 ± 0.06	0.00 ± 0.06	2.23 ± 0.08	0.01 ± 0.08
0.01	0,0,0,0,0	1.82 ± 0.11	0.01 ± 0.11	1.91 ± 0.07	0.00 ± 0.07	1.99 ± 0.08	0.01 ± 0.08	2.14 ± 0.06	0.00 ± 0.06	2.23 ± 0.08	0.01 ± 0.08
	.01,0.1,0.1,0.1,0.1	1.82 ± 0.11	0.01 ± 0.11	1.91 ± 0.07	0.00 ± 0.07	1.99 ± 0.08	0.01 ± 0.08	2.14 ± 0.06	0.00 ± 0.06	2.23 ± 0.08	0.01 ± 0.08
	0.1,0.1,0.1,0.1,0.1	1.82 ± 0.11	0.01 ± 0.11	1.92 ± 0.07	0.00 ± 0.07	1.99 ± 0.08	0.01 ± 0.08	2.15 ± 0.06	0.00 ± 0.06	2.24 ± 0.08	0.01 ± 0.08
0.1	0,0,0,0,0	1.83 ± 0.11	0.01 ± 0.11	1.92 ± 0.07	0.00 ± 0.07	1.99 ± 0.08	0.01 ± 0.08	2.15 ± 0.06	0.00 ± 0.06	2.24 ± 0.08	0.01 ± 0.08
	0.1,0.1,0.1,0.1,0.1	1.83 ± 0.11	0.01 ± 0.11	1.93 ± 0.07	0.00 ± 0.07	2.01 ± 0.08	0.01 ± 0.08	2.16 ± 0.06	0.00 ± 0.06	2.25 ± 0.08	0.01 ± 0.08
	1,1,1,1,1	1.94 ± 0.12	0.01 ± 0.12	2.04 ± 0.07	0.00 ± 0.07	2.12 ± 0.08	0.01 ± 0.08	2.25 ± 0.06	0.00 ± 0.06	2.34 ± 0.08	0.01 ± 0.08
1	0,0,0,0,0	1.95 ± 0.12	0.01 ± 0.12	2.06 ± 0.07	0.00 ± 0.07	2.14 ± 0.08	0.01 ± 0.08	2.26 ± 0.06	0.00 ± 0.06	2.35 ± 0.08	0.01 ± 0.08
	0.1,0.1,0.1,0.1,0.1	1.98 ± 0.11	0.01 ± 0.11	2.07 ± 0.07	0.00 ± 0.07	2.16 ± 0.08	0.01 ± 0.08	2.29 ± 0.04	0.00 ± 0.04	2.36 ± 0.09	0.01 ± 0.09
	1,1,1,1,1	2.11 ± 0.11	0.01 ± 0.11	2.20 ± 0.07	0.00 ± 0.07	2.29 ± 0.09	0.01 ± 0.09	2.41 ± 0.05	0.00 ± 0.05	2.47 ± 0.08	0.01 ± 0.08

## References

- [1] Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. 2022. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Adv. Neural Inf. Process. Syst.* 35 (2022), 29677–29690.
- [2] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. arXiv:1812.01097.
- [3] Sebastian Caldas, Jakub Konečný, H. Brendan McMahan, and Ameet Talwalkar. 2018. Expanding the reach of federated learning by reducing client resource requirements. arXiv:1812.07210. Retrieved from <https://arxiv.org/abs/1812.07210>.
- [4] Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. 2022. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI) Main Track*.
- [5] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. 2020. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. arXiv:2010.01243. Retrieved from <https://arxiv.org/abs/2010.01243>.
- [6] Alexandra Chouldechova and Aaron Roth. 2018. The frontiers of fairness in machine learning. arXiv:1810.08810. Retrieved from <https://arxiv.org/abs/1810.08810>.
- [7] Sen Cui, Weishen Pan, Jian Liang, Changshui Zhang, and Fei Wang. 2021. Addressing algorithmic disparity and performance inconsistency in federated learning. *Adv. Neural Inf. Process. Syst.* 34 (2021), 26091–26102.
- [8] Rong Dai, Li Shen, Fengxiang He, Xinmei Tian, and Dacheng Tao. 2022. DisPFL: Towards communication-efficient personalized federated learning via decentralized sparse training. In *International Conference on Machine Learning*. PMLR, 4587–4604.
- [9] Enmao Diao, Jie Ding, and Vahid Tarokh. 2021. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *9th International Conference on Learning Representations, (ICLR'21)*.

- [10] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *J. Mach. Learn. Res.* 20, 1 (2019), 1997–2017.
- [11] Yahya H. Ezzeldin, Shen Yan, Chaoyang He, Emilio Ferrara, and A. Salman Avestimehr. 2023. Fairfed: Enabling group fairness in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 7494–7502.
- [12] Jack Goetz, Kshitiz Malik, Duc Bui, Seungwhan Moon, Honglei Liu, and Anuj Kumar. 2019. Active federated learning. arXiv:1909.12641.
- [13] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. 2021. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Adv. Neural Inf. Process. Syst.* 34 (2021), 12876–12889.
- [14] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. 2021. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Trans. Mobile Comput.* 22, 1 (2021), 191–205.
- [15] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. arXiv:1610.02527. Retrieved from <https://arxiv.org/abs/1610.02527>
- [16] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. arXiv:1610.05492. Retrieved from <https://arxiv.org/abs/1610.05492>
- [17] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report.
- [18] Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. 2010. *An Axiomatic Theory of Fairness in Network Resource Allocation*. IEEE, Los Alamitos, CA.
- [19] Yann LeCun, Corinna Cortes, and C. J. Burges. 2010. MNIST Handwritten Digit Database. ATT Labs. Retrieved from <http://yann.lecun.com/exdb/mnist>
- [20] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. 2021. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 42–55.
- [21] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*. PMLR, 6357–6368.
- [22] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Sign. Process. Mag.* 37, 3 (2020), 50–60.
- [23] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. 2019. Fair resource allocation in federated learning. arXiv:1905.10497. Retrieved from <https://arxiv.org/abs/1905.10497>
- [24] Xiaoli Li, Siran Zhao, Chuan Chen, and Zibin Zheng. 2023. Heterogeneity-aware fair federated learning. *Inf. Sci.* 619 (2023), 968–986.
- [25] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Adv. Neural Inf. Process. Syst.* 33 (2020), 2351–2363.
- [26] Lingjuan Lyu, Xinyi Xu, Qian Wang, and Han Yu. 2020. Collaborative fairness in federated learning. *Fed. Learn.: Priv. Incentive* (2020), 189–204.
- [27] Lingjuan Lyu, Jiangshan Yu, Karthik Nandakumar, Yitong Li, Xingjun Ma, Jiong Jin, Han Yu, and Kee Siong Ng. 2020. Towards fair and privacy-preserving federated deep models. *IEEE Trans. Parallel Distrib. Syst.* 31, 11 (2020), 2524–2541.
- [28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.
- [29] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Comput. Surv.* 54, 6 (2021), 1–35.
- [30] Hamid Mozaffari and Amir Houmansadr. 2022. E2FL: Equal and equitable federated learning. arXiv:2205.10454. Retrieved from <https://arxiv.org/abs/2205.10454>
- [31] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*. PMLR, 4095–4104.
- [32] Bozidar Radunovic and Jean-Yves Le Boudec. 2007. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Trans. Network.* 15, 5 (2007), 1073–1083. <https://doi.org/10.1109/TNET.2007.896231>
- [33] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2021. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.* 54, 4 (2021), 1–34.
- [34] Yuxin Shi, Han Yu, and Cyril Leung. 2023. Towards fairness-aware federated learning. *IEEE Trans. Neural Netw. Learn. Syst.* (2023).
- [35] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing federated learning on non-iid data with reinforcement learning. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'20)*. IEEE, 1698–1707.

- [36] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. 2021. A novel framework for the analysis and design of heterogeneous federated learning. *IEEE Trans. Sign. Process.* 69 (2021), 5234–5249.
- [37] Su Wang, Mengyuan Lee, Seyyedali Hosseinalipour, Roberto Morabito, Mung Chiang, and Christopher G Brinton. 2021. Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'21)*. IEEE, 1–10.
- [38] Zheng Wang, Xiaoliang Fan, Jianzhong Qi, Chenglu Wen, Cheng Wang, and Rongshan Yu. 2021. Federated learning with fair averaging. arXiv:2104.14937. Retrieved from <https://arxiv.org/abs/2104.14937>
- [39] Dezhong Yao, Wanning Pan, Michael J. O’Neill, Yutong Dai, Yao Wan, Hai Jin, and Lichao Sun. 2021. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. arXiv:2111.14655. Retrieved from <https://arxiv.org/abs/2111.14655>
- [40] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. 2020. A fairness-aware incentive scheme for federated learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 393–399.

Received 22 April 2024; revised 4 September 2024; accepted 24 October 2024