國立陽明交通大學
資訊科學與工程研究所
碩士論文

Institute of Computer Science and Engineering
National Yang Ming Chiao Tung University
Master Thesis

聯邦式學習中兼顧準確率與延遲的客户機選擇

Joint Accuracy- and Latency-Optimized Client Selections for Federated Learning

研 究 生： 蔡沅恆 (Tsai, Yuan-Heng)
指導教授： 嚴力行 (Yen, Li-Hsing)

中華民國 一一三年八月
August 2024

聯邦式學習中兼顧準確率與延遲的客户機選擇

Joint Accuracy- and Latency-Optimized Client Selections for Federated Learning

研 究 生： 蔡沅恆　　Student： Yuan-Heng Tsai

指導教授： 嚴力行 博士　　Advisor： Dr. Li-Hsing Yen

國立陽明交通大學

資訊科學與工程研究所

碩士論文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Yang Ming Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

August 2024

Taiwan, Republic of China

中華民國 一一三年八月

誌　謝

謝天謝地

XXXXX於

國立交通大學網路工程研究所碩士班

中華民國 108 年 8 月

聯邦式學習中兼顧準確率與延遲的客户機選擇

學生：蔡沅恆　　指導教授：嚴力行 博士

國立陽明交通大學 資訊科學與工程研究所 碩士班

摘　要

聯邦式學習(Federated Learning)是現今受歡迎的一種分散式學習方式。它透過每一輪傳遞模型參數，使得多個用户設備能夠與中央伺服器協作進行模型訓練。透過傳輸模型權重或梯度取代直接傳輸於用户端的訓練資料庫，聯邦式學習可以確保用户隱私同時降低減少網路負擔。然而由於本地訓練設備或數據收集者的行爲不同，設備之間存在硬體和數據異質性，這導致非獨立同分佈（non-IID）數據和拖尾效應（straggler effect）產生，期會造成訓練效能降低以及訓練時間更長。因此，更有策略的訓練者選擇(client selection) 至關重要。在本論文中，我們聚焦於同時存在硬體和數據異質性訓練者環境下，透過訓練者選擇(client selection)優化訓練效能和收斂延遲。爲了應對由資訊不對稱帶來的挑戰，我們首先提出了一種隱私保護的方法來量化每個設備上的非獨立同分佈程度。接著，我們提出了結合Shapley值與嶺迴歸模型(ridge regression)之方法來預測每個訓練者對準確性的貢獻。最終，透過同時考量每個訓練者的準確度預測值與訓練延遲決定出最佳訓練者集合進行訓練。此外，我們進行了實驗，結果顯示基於Shapley值的準確性預測模型能夠反應眞實測量準確度。接著我們在不同的非獨立同分佈程度下與四種不同演算進行比較對，結果證明了我們方法的有效性以及較好的表現。

聯邦式學習、資料與硬體異質性、**Shapley**值、嶺迴歸模型

Joint Accuracy- and Latency-Optimized Client Selections for Federated Learning

Student : Yuan-Heng Tsai Advisor: Dr. Li-Hsing Yen

Institute of Computer Science and Engineering
National Yang Ming Chiao Tung University

Abstract

Federated Learning (FL) is a popular distributed learning algorithm that operates by iteratively performing local training on devices and aggregating the locally updated models or gradients on a central server. FL enables training across decentralized devices while ensuring privacy and reducing network overhead by transmitting model weights or gradients instead of raw data. However, data and hardware heterogeneity exists among these devices due to the varying behaviors of local trainers or data collectors, which lead to performance degradation due to non-independent and identically distributed (non-IID) data and the straggler effect, which results in longer training latency. Therefore, intelligent client selection is crucial. In this thesis, we focus on the challenge of jointly optimizing accuracy and convergence latency through client selection, while considering both hardware and data heterogeneity. To address the challenges posed by asymmetric information, we propose a privacy-aware method to quantify the non-IID level in each device first. Then we introduce a Shapley value-based method combined with a ridge regression model to predict each client's contribution to accuracy. Finally, we solve the problem by incorporating both predicted accuracy and latency. Additionally, we conducted experiments to show that the Shapley-based accuracy predictor achieves a small gap between prediction and actual evaluation and then evaluated our algorithm under different non-IID levels, comparing it

to four benchmarks. The results demonstrated the effectiveness of our approach.

federated learning, data & hardware heterogeneity, Shapley value, ridge regression.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

In recent years, machine learning (ML) has experienced unprecedented growth and innovation, fundamentally transforming various sectors from healthcare to finance. The advent of deep learning, characterized by sophisticated neural network architectures, has been a major driving force behind this rapid development. Breakthroughs such as AlexNet [1] in 2012, which significantly improved image recognition performance, showcased the potential of deep neural networks and spurred further research and development.

Despite the significant success of centralized machine learning, training data often relies on collections from edge devices, such as IoT or mobile devices, which present several challenges that hinder its practical application. Growing privacy concerns, such as those outlined in the General Data Protection Regulation (GDPR) [2], have made data sharing increasingly challenging. This prevents centralized task owners from collecting sensitive data from users. Additionally, transmitting large amounts of raw data over the Internet introduces substantial network overhead issues. These challenges necessitate a shift towards distributed learning, where learning occurs on local devices, and only the updated model or gradients are transmitted to the server instead of the raw data.

In response to these challenges, McMahan et al. proposed a revolutionary approach to privacy-preserving machine learning known as Federated Learning (FL) [3], which enables the training in decentralized devices to ensure privacy and alleviate network overhead. It operates

by iteratively performing local training on devices and aggregating the locally updated models or gradients on a central server. The process begins with the central server initializing a global model and sending it to all participating devices. Each device then uses its local data to train the model and compute updates, such as gradients or model parameters. These locally computed updates are sent back to the central server, which aggregates them by weighted averaging based on the size of the local datasets to update the global model. The updated global model is then redistributed to the devices, and the process repeats until convergence.

There are several issues associated with federated learning. Due to the different behaviors of local trainers or data collectors, there are varying data distributions among clients, including differences in label distribution, feature distribution, quantity distribution, and mislabeled datasets. This phenomenon, known as non-independent and identically distributed (non-IID) data, can degrade the performance of the global model and increase convergence latency, particularly from participants with severely non-IID datasets. Additionally, client hardware heterogeneity can lead to a straggler effect due to varying computation and transmission capacities. In this scenario, the slowest client delays the entire learning process because the server must wait for its model for model aggregation, even if all other clients have completed their training quickly. Furthermore, battery constraints must also be considered for a more comprehensive algorithm for federated learning on commercial mobile devices or IoT devices.

In this work, we aim to design a client selection mechanism for federated learning with battery-constrained local devices, considering both accuracy and convergence latency. The primary distinction of our approach compared to previous studies is our comprehensive handling of the non-IID (non-independent and identically distributed) data challenge, whereas most prior research focuses on only specific aspects of the non-IID scenario. Finally, we present evaluation experiments that demonstrate our algorithm's superior generalization capability across various

non-IID settings, along with a reduced need for hyperparameters compared to other algorithms.

In this thesis, Chapter 2 introduces the background and related studies of FL. Chapter 3 models our system, detailing the specific FL settings in our study and formulating the computation transmission latency and energy consumption. Chapter 4 formulates the problem of maximizing accuracy while minimizing training latency, converting the problem from an offline to an online format. Before addressing the problem defined in Chapter 4, we conducted several experiments with results presented in Chapter 5 to understand how non-IID data impacts FL performance and convergence. We also propose a method to measure the non-IID level for each client. In Chapter 6, we introduce a Shapley-based method combined with a ridge regression model to predict each client's contribution to accuracy. Then, we solve the problem by incorporating both predicted accuracy and latency. Finally, Chapter 7 shows the results of experiments conducted to evaluate our algorithm under different non-IID levels with comparisions to four benchmarks, demonstrating the effectiveness of our approach.

Chapter 2. Background and Related Work

In the following section, we survey several studies on different aspects of FL, including general surveys, optimization of FL frameworks, data heterogeneity issues, and hardware heterogeneity issues.

2.1 Federated Learning

Federated Averaging (FedAvg) is a foundational algorithm in FL introduced by McMahan et al. in 2017. This algorithm has become the cornerstone for many studies in the field, defining a framework that involves two main roles: the task owner, referred to as the server, and the data owners, referred to as clients. The server is responsible for initiating the learning task and managing the entire FL workflow, which includes selecting clients and aggregating their models. The clients, on the other hand, are numerous devices that possess private datasets and perform local training on these datasets. Due to transmission overhead and budget constraints, only a subset of clients is selected to participate in training during each round. This selective participation helps balance the need for effective learning with the practical limitations of network and resource usage.

The FL training process consists of multiple global rounds, each involving several local training epochs by a subset of clients. Here are the detailed steps of each round

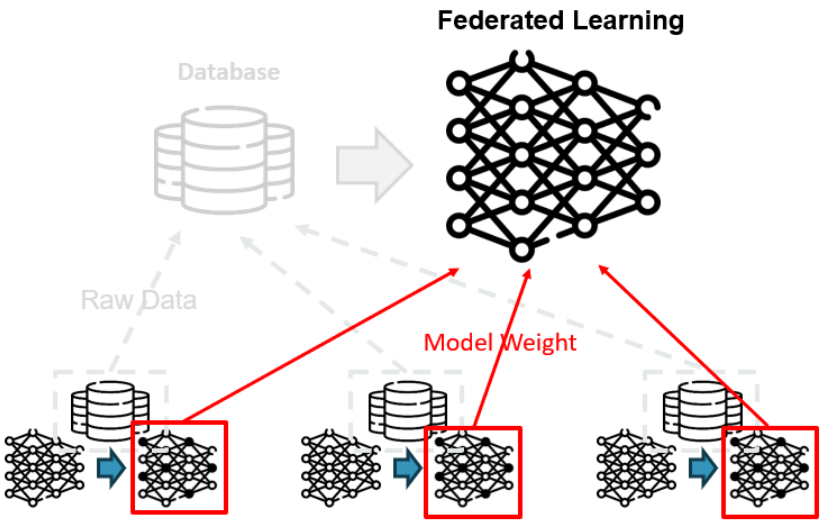


Figure 1: Federated Learning

1) **Learning Task Initialing**: The server initiates the learning task and recruits voluntary clients. Typically, this task involves training a neural network (NN)-based machine learning model.

2) **Client Selection**: In each training round, the server selects a subset of clients to participate in the training. Commonly, the server randomly selects clients without considering their data or hardware heterogeneity.

3) **Model Downloading & Local Training**: After determining the selected clients, the server broadcasts the current global model to them. These clients then perform local training with the required learning algorithm. For the NN-based machine learning task, the training is usually conducted via Stochastic Gradient Descent (SGD) with a specified number of local epochs I_{loc}, a training batch size B_{loc}, and a local training learning rate γ_{loc}.

4) **Model Uploading**: After completing a round of local training, the clients upload their locally trained model weights back to the server.

5) **Model Aggregation**: Upon receiving the updated models from the clients, the server

aggregates them. The common aggregation method is the FedAvg algorithm, where local models are aggregated using weights based on the clients' dataset sizes.

$$\omega' = \frac{\sum_{i \in C} |D_i| \omega_i}{\sum_{i \in C} |D_i|}, \tag{2.1}$$

where C is the subset of clients that conducted local training, ω_i is the updated local model from client i, ω' is the updated global model, and the $|D_i|$ is the dataset size of the client i.

6) **Repeat Steps 2-5 until reaching convergence condition**: The process is repeated from steps 2 to 5 until the convergence condition is reached.

The study [4] provides the different update methods. Instead of using first-order gradient descent (GD) to update the local model, momentum gradient descent (MGD) is used to get better convergence performance. The study [5] proposed Agnostic Federated Learning, a new framework that optimizes the centralized model for any target distribution formed by a mixture of the client distributions, and it showed that this framework naturally yields a notion of fairness. Besides the popular horizontal FL framework, the study [6] [7] discusses Vertical Federated Learning (VFL), which is available in the case that the user features of the two datasets overlap little, but the users overlap a lot. The studies [8] [9] [10] [11] [12] [13] proposed the theory to model the relation between convergence time and convergence condition on L-Lipschitz and γ -strongly convex loss function. To quickly overview the FL concept, challenge, category, and application, the studies [14] [15] [16] [17] provided the comprehensive survey on the challenge, category, topic, and application on federated learning.

2.2 Data Heterogeneity

In federated learning, the data heterogeneity, or independent and identically distributed(non-IID), is caused by the differences in behavior and environment of the data collector and local datasets. There are a lot of studies indicating that non-iid makes the model performance and convergence performance decline. From the studies [18] [19] [20], the non-IID can fall into several classes, feature skew, label skew, and quantity skew. The feature skew includes the difference between each client's feature distribution and the feature corresponding to different labels. The label skew is the difference between each client's label distribution, and the quantity skew is the different amount of the dataset size between each client. The study [21] conducted an experiment on the Medical Imaging dataset with heterogeneous data and shows the performance degrades with the increasing degrees of data heterogeneity. The study [22] shows how feature skew, label skew, and quantity skew will affect the performance of state-of-the-art federated learning algorithms. Here, we provide a detailed survey of studies based on various types of non-IID data:

2.2.1 Quantity Skew

In centralized machine learning, studies such as [23], [24], [25], [26], and [27] provide both theoretical analyses and experimental evidence showing that the size of the training dataset significantly affects training outcomes, especially in image classification tasks. When the dataset samples are insufficient, the trained model tends to overfit, resulting in poor test performance.

In distributed learning scenarios, particularly in federated learning, the size of the data not only impacts the generalizability of the trained model but also affects its convergence speed. Studies like [8], [9], [10], [11], [12], and [13] explore the relationship between convergence time

and convergence conditions based on L-Lipschitz and γ -strongly convex loss functions. These studies identify dataset size as a factor that can accelerate model convergence. Consequently, they incorporate data size into their problem formulations and create incentives to select clients with larger datasets to enhance efficiency.

Further emphasizing this point, the studies [28] and [29] highlight that dataset size influences both the convergence speed and the energy consumption during training. They frame the dataset size of each client as an optimization problem to identify the optimal trade-off between energy efficiency and training latency.

2.2.2 Label Skew

Concerning label skew, the study [20] provides a theoretical analysis demonstrating that weight divergence during training is constrained by the Earth Mover's Distance (EMD) between the class distributions on each device (or client) and the overall population distribution. They highlight how an excessive label distribution can cause significant model weight divergence. The study [20] explores label distribution skew from a statistical perspective, theoretically and empirically showing that traditional methods relying on softmax cross-entropy are inadequate. To address this, they introduce an algorithm that adjusts the logits before softmax cross-entropy based on the occurrence probability of each class at the local update step, which enhances performance in scenarios with label skew. Additionally, to mitigate the challenges associated with imbalanced data distributions, the studies [30] and [31] recommend augmenting the local dataset with extra samples to improve model performance. To further address label skew, various approaches have been proposed to prevent performance degradation due to the aggregation of highly divergent models. Personalized Federated Learning (PFL), as proposed in [32], [33], [34], [35], and [36], adds personalized layers to each client's local model. These

layers are not aggregated, ensuring that only unpersonalized layers are updated on the server side. Clustered Federated Learning, detailed in [37], [38], [39], [34], and [38], clusters clients with similar metrics at the beginning or during learning, aggregating only models from the same cluster. In a different approach, studies [40], [41], [42], [43], and [44] determine the type and number of learning models before starting and select the most beneficial clients to participate in various learning tasks. Conversely, to get multiple models the results, studies [45], [46], [47] and [48] determine aggregate weights not by the dataset size of selected clients but by the label-skew level. This approach reduces the impact of low-quality clients by lowering their weight in the aggregation step. Finally, DRL-based approaches for client selection, as discussed in [49], [50], [51], [52], [53], [54], [55], [56], and [57], use deep reinforcement learning to optimize client selection in the face of heterogeneous data and label distributions.

2.2.3 Mislabeling

In real-world environments, due to malicious client attacks (known as poison attacks), insufficient computational capabilities of clients, and inexperienced labeling (known as noisy labels), local datasets may contain mislabeled data, which can degrade the performance of the global model. Studies [58] and [59] conducted experiments on mislabeling flipped by malicious clients, showing that mislabels can severely impact performance, thus highlighting the importance of addressing mislabels. for deliberate poison attacks, studies [59] and [58] use updated gradients to detect such activities. [58] notes that since attackers have specific targets rather than random, the gradients submitted by attackers will update in a direction different from the real target; hence, the cosine similarity between the local update gradient and the global update gradient is used as a metric to measure the extent of mislabeling. On the other hand, there are several studies to overcome the performance degrading caused by noisy labels; methods in stud-

ies [60], [61], and [62] primarily focus on detecting and filtering out mislabeled samples from local datasets. Studies [63], [64], and [65] mitigate the performance degradation by adding additional information to the local dataset. The studies [58], [59], [66], [67], [68], and [69] identify clients that may contain mislabels and then mitigate the impact of mislabeled clients by adjusting the aggregation weights. The studies [68], [54], and [69] assume complete trust from clients and use evaluations from pre-trained models on local models to measure mislabel levels.

2.3 Impact of Hardware Heterogeneity

In federated learning, dataset heterogeneity and variations in hardware resources and communication environments among clients lead to significant challenges. These variations affect availability, computational speed, and transmission speed, reducing convergence speed and overall model performance. The study [70] offers a comprehensive survey that outlines the critical heterogeneity challenges federated learning must address to be applicable in real-world scenarios. This survey emphasizes computational heterogeneity among devices, providing a detailed review of recent advancements in heterogeneity-aware federated learning. Another pivotal study, [71], investigates the impacts of heterogeneous hardware specifications and dynamic states on the FL process. This groundbreaking research, based on data from 136,000 smartphones and utilizing state-of-the-art FL algorithms, marks the first large-scale study of its kind. The findings reveal that heterogeneity can cause accuracy to drop by up to 9.2% and increase convergence time by as much as 2.32 times. Addressing these concerns, a substantial body of research, including studies from [46], [72], [73], [47], [50], [51], [52], [53], [74], [56], and [57], has been dedicated to developing strategies to minimize latency. These studies are instrumental in enhancing the efficiency of federated learning systems by ensuring quicker model convergence across diverse network conditions. Furthermore, the energy consumption associated with

client training is a critical consideration. Research efforts from [28], [29], [46], [72], [47], [50], [52], [74], [56], [57], and [75] explore methods for minimizing energy use during the federated learning process. Moreover, the limited battery size of clients restricts client selection times, making battery capacity a significant consideration in IoT or mobile device environments. The studies [29], [72], [53], and [57] take into account these battery constraints.

2.4 Fairness Concerns

In previous studies addressing hardware and data heterogeneity, online strategies are typically employed, often leading to preferential selection of specific clients. This uneven selection can cause the global model to predominantly learn from a subset of clients, resulting in rapid convergence to a local optimum and subsequent degradation in overall performance. Therefore, ensuring that all clients have multiple opportunities to participate can significantly enhance model performance. It is commonly referred to as fairness. The study [76] provides a comprehensive survey of fairness-aware federated learning (FAFL), presenting a state-of-the-art taxonomy of FAFL approaches. These cover crucial FL processes, including client selection, optimization, contribution evaluation, and incentive distribution. Additionally, the study discusses key metrics for experimentally evaluating FAFL approaches and suggests promising directions for future research. [77] notes that overly frequent selection of more stable clients can introduce bias, thus degrading training effectiveness. To address this, they propose an Exponential Weight Algorithm for Exploration and Exploitation (Exp3)-based client selection method (E3CS), which considers both effective participation and fairness. Meanwhile, [48] adjusts the aggregate weight for each selected client based on the quantity of information from training accuracy and frequency. The studies [78], [79], [75], and [80] implement long-term fairness constraints to ensure that the average participation rate of each client meets or exceeds a guar-

anteed expected rate. They employ Lyapunov optimization to transform the original offline problem into an online optimization challenge.

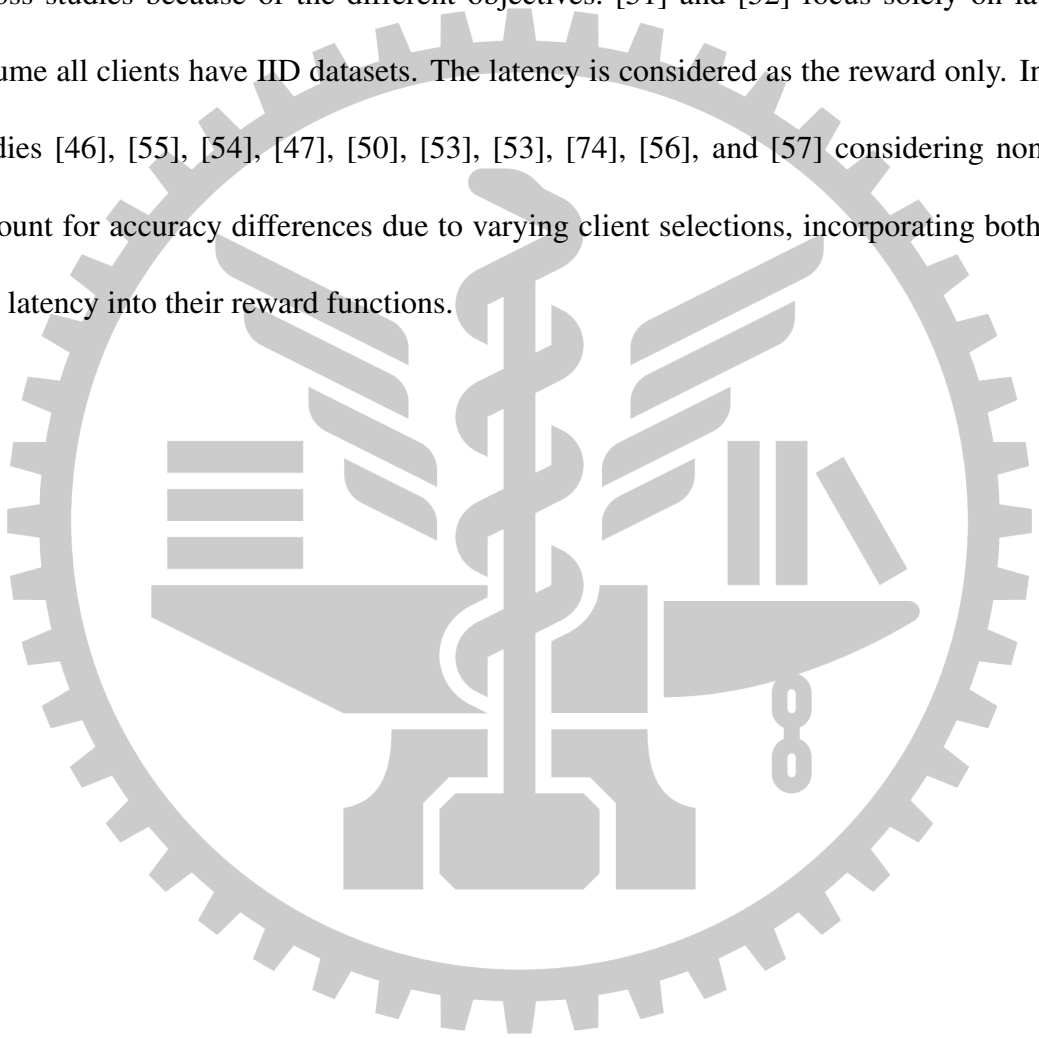
2.5 Client Selection Approaches

Considering hardware and data heterogeneity, client selection is crucial for maximizing multiple objectives in federated learning by selecting the best clients for each training round. Several studies, including [75], [28], [8], [9], [10], [11], [12], and [13], have addressed this problem using traditional optimization methods. Additionally, some research has applied game theory to solve the client selection problem with self-interest considerations. For example, [72] uses matching theory to find the best client set with the lowest latency and highest contribution. [81] uses the Stackelberg Game to model the problem as a sequential decision-making process, where the server and clients make decisions sequentially, affecting future choices.

Optimized solutions like multi-armed bandit (MAB) and reinforcement learning (RL) have also been proposed for selecting clients because of the challenge of modeling the relation between the different factors and the training performance for the traditional optimization solution and the complexity consideration, especially considering the hardware and data heterogeneity at the same time. Studies such as [79] and [82] consider both fairness and latency. They use the MAB model to predict client training latency and model fairness as part of the objective, which is solved using Lyapunov optimization. In [83], latency and accuracy are incorporated into the MAB's reward function, motivating the server to select clients with lower latency and higher-quality datasets. [84] employs Contextual Multi-Armed Bandits (CMAB), which enhances the selection mechanism by considering historical feedback and available action information.

For more complex learning mechanisms, reinforcement learning (RL) models are used to select clients in studies like [51], [52], [46], [55], [54], [47], [50], [53], [74], [56], and [57].

RL updates a neural network-based model using historical actions, states, and rewards. These studies consider latency and model performance, including non-IID data or client quality. For instance, [54] evaluates clients' datasets before training to obtain metrics representing dataset quality, which are used as the client state. Studies like [52], [53], [74], [56], and [57] use local model weights or updated gradients as the state. Moreover, the reward design varies across studies because of the different objectives. [51] and [52] focus solely on latency and assume all clients have IID datasets. The latency is considered as the reward only. In contrast, studies [46], [55], [54], [47], [50], [53], [53], [74], [56], and [57] considering non-IID data account for accuracy differences due to varying client selections, incorporating both accuracy and latency into their reward functions.



Chapter 3. System Model

In our study, as shown in Figure 2, we discuss FL within a network composed of both a wireless network and a core network. The server is assumed to be directly connected to the core network, possessing strong computational capacity and a sufficient energy supply, allowing us to ignore the computation latency at the server. The server contains a testing dataset, denoted as D_g which is used to evaluate the global model at each round t by

$$\text{acc}(\omega^{t+1}, D_g) = \frac{\sum_{(x_i,y_i)\in D_g} I(F(x_i, \omega^{t+1}) = y_i)}{\mid D_g \mid}, \tag{3.1}$$

where the $F(x_i, \omega^{t+1})$ is the function that returns the predicted label of the sample x_i under model weight ω^{t+1} at round $t+1$ and I is an indicator function that returns one if $F(x_i, \omega^{t+1}) = y_i$ (i.e., the prediction is correct) and zero otherwise. It is important to note that the size of the testing dataset is smaller than the necessary size for learning a robust model, and there is no mislabeling in the testing dataset.

Our study focuses on IoT and mobile environments where the client devices are constrained by limited battery life and hardware capacity. The clients in our study possess datasets with various non-IID features, including label skew, quantity skew, feature skew, and mislabeling. We denote the dataset of client i as D_i and its size as $|D_i|$. In addition to the variations in dataset features, each client has different hardware capabilities. Each client can only recharge their

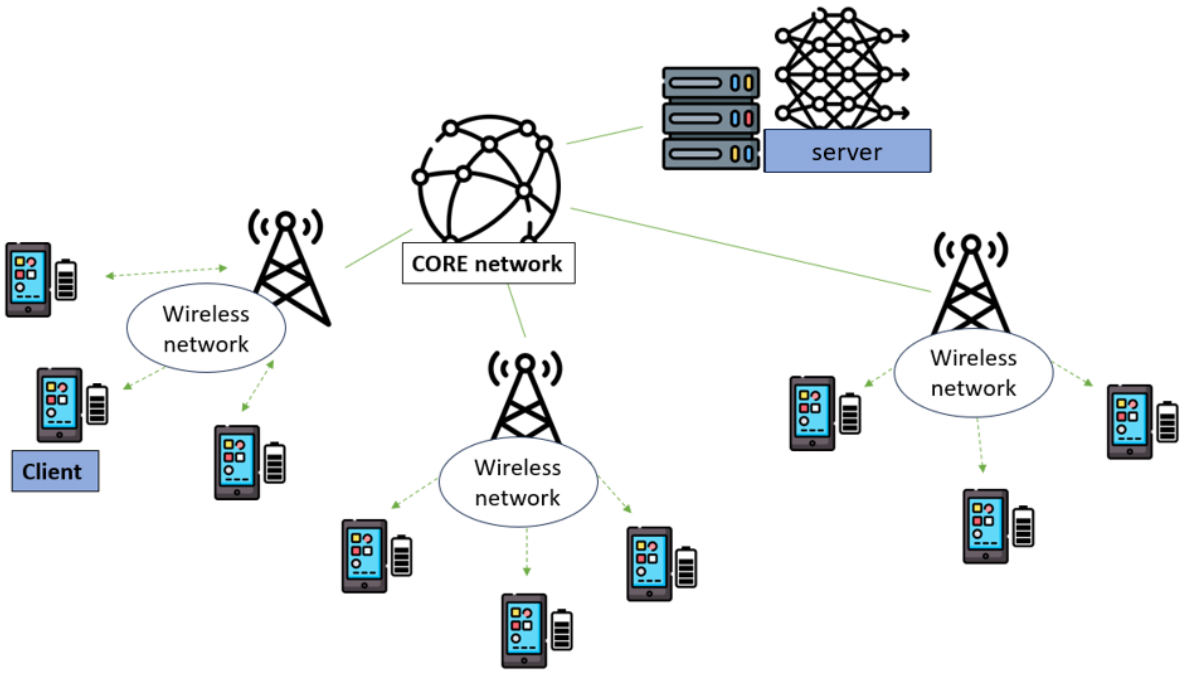


Figure 2: system architecture

batteries once the training task is completed. If a client depletes its battery by participating in the training, it will not be selected again. Additionally, clients have varying computational and transmission capacities, which lead to differences in energy consumption and training latency.

Due to battery constraints and unstable network conditions, the set of available clients at each round t varies, denoted as $C^t_{\text{ava}} \in C$, where C is the set of all clients. Instead of random selection, the server strategically selects the client c_t from C^t_{ava}, considering the client's dataset quality and hardware capacity to maximize the objectives. To simplify the problem, the size of the selected subset in each round, $|C^t|$, is fixed and denoted as N_T. In our specific study, N_T is set to 10. Additionally, to prevent endless waiting for failed clients during the aggregation stage, there is a maximum allowable latency l_{max}. If a client's local training latency l^t_i exceeds l_{max}, its training result is discarded, and the set of successfully uploaded clients is denoted as C^t_{suc}. In our system model, we adopt the same aggregation method as in FedAvg to aggregate the updated local model from client i, ω^t_i, into the updated global model ω^{t+1}.

In the following section, we model the computation and transmission latency and energy in

more detail.

3.1 Computation Latency and Energy Consumption

The selected clients update the model weights in each local training round using their CPU or GPU. Several variables are considered to model the latency and energy consumption of their computation units. For each client, the computation frequency f_i^t varies over time because multiple processes may run concurrently on the same device, affecting the training's ability to run at full capacity. This work assumes that the computation frequency is fixed during the duration of local training. The device heterogeneity coefficient c_i represents the varying computational capacities of each client's device with different task types. Then the computation latency $l_{\text{cmp},i}^t$ of client i at round t is calculated as:

$$l_{\text{cmp},i}^t = \frac{c_i I_{\text{loc}} |D_i| s}{f_i^t}, \tag{3.2}$$

where I_{loc} is the number of local epochs, $|D_i|$ is the dataset size of client i, and s is the sample size. The computation energy for client i at round t, denoted as $e_{\text{cmp},i}^t$, is calculated as:

$$e_{\text{cmp},i}^t = k I_L c_i D_i f_i^{t2}, \tag{3.3}$$

where k is the energy coefficient that describes the device's energy consumption, which depends on the chip architecture.

3.2 Transmission Latency and Energy Consumption

The communication latency $l^t_{\text{com},i}$ for client i at round t consists of several components: latency from the wireless network $l^t_{\text{wireless},i}$, latency from the Radio Access Network (RAN) $l^t_{\text{RAN},i}$, and latency from the core network $l^t_{\text{core},i}$.

The latency in the RAN $l^t_{\text{RAN},i}$, is influenced by various factors related to the connected gNB settings. These include numerology, slot duration, scheduling, transmission mode, allocated bandwidth, and the density of UE devices, among others. In this work, we reference the study by [85], which provides a comprehensive model that delineates the relationships between these factors and the resultant latency in the $l^t_{\text{RAN},i}$. The latency arising in the core network $l^t_{\text{core},i}$ includes transit delays, propagation delays, and transmission latency to the internet. The paper by [86] models the core network as an M/M/1 queuing model. For simplicity and to conserve space in our presentation, we denote them collectively as $l^t_{\text{RAN},i}$ and $l^t_{\text{core},i}$.

The latency in the wireless network $l^t_{\text{wireless},i}$ can be calculated using the Shannon-Hartley theorem. This theorem provides the achievable transmission rate λ^t_i for client i at round t in a wireless network, which is calculated as

$$\lambda^t_i = B^t_i \ln(1 + \frac{g^t_i p^t_i}{N^2}) \text{ (in bit/s)}, \tag{3.4}$$

where the b^t_i represents the allocated bandwidth for client i at round t, g^t_i is the channel gain between the client and its base station, p^t_i is the transmission power of client i (in Watts/Hz), and N is the Gaussian noise power. Then the latency in wireless network $l^t_{\text{wireless},i}$ can be calculated

with the learning model size $|\omega_i^t|$ (in bits) and achievable transmission rate λ_i^t by

$$l^t_{\text{wireless},i} = \frac{|\omega_i^t|}{\lambda_i^t} \text{ (in bit/s).} \tag{3.5}$$

The total latency of client i at round t is:

$$l^t_{\text{com},i} = l^t_{\text{wireless},i} + l^t_{\text{RAN},i} + l^t_{\text{core},i}\ . \tag{3.6}$$

The transmission energy consumption of client i at round t, $e^t_{\text{com},i}$ (in watts) is the energy consumed by the client to transmit data through the wireless network and calculated by

$$e^t_{\text{com},i} = p_i^t l^t_{\text{wireless},i}\ . \tag{3.7}$$

3.3 Overall Latency and Energy Consumption

After modeling the transmission model and computation model of the client device, then we can calculate the latency(l_i^t) of the client i at round t as the computation latency adds the transmission latency if the client i participate the training at round t, else get 0

$$l_i^t = \begin{cases} l^t_{\text{cmp},i} + l^t_{\text{com},i} & \text{if client } i \text{ participates training at round } t, \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

The latency at round t, l^t is determined by the maximum training latency among the selected clients. A maximum waiting time l_{max} is set to avoid unlimited waiting due to disconnected or failing clients. If the server's waiting time exceeds l_{max}, then any unuploaded data is discarded.

So the L_t can be formular as

$$l^t = \max_{i \in C} \min(l_{\max}, l_i^t) \ . \tag{3.9}$$

For energy consumption e_i^t of client i at round t, in addition to considering the energy consumed by computation and transmission, additional energy consumption is associated with keeping the device activated and running other applications on the device. This is denoted as $e_{\text{nom},i}^t$, which exists no matter if the client participates at round t, and it occurs regardless of whether the client participates in the training round t

$$e_i^t = \begin{cases} e_{\text{cmp},i}^t + e_{\text{com},i}^t & \text{if client } i \text{ participates training at round } t \\ e_{\text{nom},i}^t & \text{otherwise.} \end{cases} \tag{3.10}$$

The remained battery b_i^t of client i at end of round t is the remained battery at round t subtract the energy consumption at round t:

$$b_i^t = b_i^{t-1} - e_t^i. \tag{3.11}$$

We list the notation in Table 1

Table 1: Notation

Notation	Discription
s	sample size
I_{loc}	local training local epoch
B_{loc}	local training batch size
R	required global round
l_{max}	maximum allowable latency
ω^t	global model weight at the beginning of the round t
ω_i^t	updated Local model weight at the round t
C	the client set
C_{ava}^t	available client set at the round t
C^t	selected client set at the round t
C_{suc}^t	The set of clients that successfully upload their model weights
D_i	the client i's dataset
D_{g}	server's test dataset
c_i	the client i's device heterogeneity coefficient
f_i^t	the client i's device computation frequency at the round t
k_i	the client i's device energy coefficient
B_i^t	the client i's allocated bandwidth at round t
g_i^t	the client i's channel gain between the client and its base station at round t
N_i	the client i's Gaussian noise power
p_i^t	the client i's transmission power at the round t
λ_i^t	the client i's achievable transmission rate at the round t
l^t	the latency at the round t
l_i^t	the client i's latency at the round t
$l_{\text{cmp},i}^t$	the client i's computation latency at the round t
$l_{\text{com},i}^t$	the client i's transmission latency at the round t
$l_{\text{wireless},i}^t$	the client i's latency from the wireless network at the round t
$l_{\text{RAN},i}^t$	the client i's latency from the RAN at the round t
$l_{\text{core},i}^t$	the client i's latency from the core network at the round t
e_i^t	the client i's energy consumption at the round t
$e_{\text{cmp},i}^t$	the client i's computation energy consumption at the round t
$e_{\text{com},i}^t$	the client i's transmission energy consumption at the round t
b_i^t	remained battery at the round t
$e_{\text{nom},i}^t$	the client i's maintaining and others application energy consumption at the round t

Chapter 4. Problem Formulation

In our study, we aim to maximize the test accuracy of the model while minimizing training latency in FL to achieve optimal convergence. The balance between training latency and test accuracy can be regulated by an importance weight δ. Data and hardware heterogeneity pose significant challenges; for instance, methods like FedAvg, which randomly select clients without assessing their performance, often result in decreased model performance and increased latency due to low-quality data and the straggler effect. To address this issue, we formulate the problem of selecting clients in a way that simultaneously optimizes performance and convergence latency. then, the problem is formulated as

$$\text{P1}: \max_{\{A_1, A_2, ..., A_T\}} \text{acc}(\omega_T) - \delta \sum_{t=1}^{T} l^t, \tag{4.1}$$

$$\text{s.t.} \quad a_i^t \in \{0, 1\}, \forall i, t \tag{4.2}$$

$$a_i^t = 0, \forall i \notin C_{\text{ava}}^t, t \tag{4.3}$$

$$\sum_{i \in C} a_i^t = \min(N_T, |C_{\text{ava}}^t|), \forall t \tag{4.4}$$

$$a_i^t e_i^t \leq b_i^t, \forall i, t \tag{4.5}$$

In the formula, A_t represents the client selection at round t, and it is the $|C|$ dimension vector $A_t = \{a_1^t, a_2^t, ..., a_{|C|}^t\}$, where $a_i^t \in \{0, 1\}$ is an indicator. When a_i^t is 1, the client i is selected to participate in the training at round t; otherwise, the client does not participate during that

round. The constraint (4.3) indicates that only the available client at round t could be selected. The constraint (4.5) is the battery-aware constraint indicating that a client whose battery is lower than or equal to the energy consumption will not be selected.

Although the transmission model outlined in Chapter 3 is comprehensive and accurately models transmission latency, there remains a challenge. In commercial or standard networks, control parameters are managed by mobile network operators and are typically inaccessible to regular network users. Furthermore, these parameters are subject to dynamic changes due to unpredictable fluctuations in network conditions, such as variations in user density at connected base stations or alterations in routing paths. Additionally, modern mobile devices, which perform multiple tasks, have their resources allocated by the operating system. The unpredictable nature of incoming tasks and uncontrolled CPU scheduling complicate the energy consumption prediction, contributing to a significant discrepancy between theoretical optimization and practical solutions, particularly for long-term predictions.

To more effectively address these challenges, it is advantageous to transform the offline problem, denoted as P1, into an online format, solving it in real time. This transformation requires focusing on the term $\sum_{t=1}^{T} l^t$, where the total latency of FL is determined by the sum of the training latency and the number of global rounds required to meet the convergence condition. Selecting clients with lower training latency can reduce the latency from the first term. Simultaneously, choosing clients that demonstrate greater accuracy progress in each round can minimize the latency from the second term. This strategy reduces the number of training rounds needed and aligns with the dual objectives of enhancing both speed and accuracy. Thus, we can

reformulate the problem from the offline format (P1) to the online format (P2) as

$$\text{P2}: \max_{A_t} \Delta\text{acc}_t - \delta l^t, \tag{4.6}$$

$$\text{s.t.} \quad a_i^t \in \{0, 1\}, \forall i, \tag{4.7}$$

$$a_i^t = 0, \forall i \notin C_a^t \tag{4.8}$$

$$\sum_{i \in C} a_i^t = \min(N_T, |C_a^t|) \tag{4.9}$$

$$a_i^t e_i^t \leq b_i^t, \forall i, \tag{4.10}$$

where Δacc_t represents the model performance progress between training rounds, calculated as $\Delta\text{acc}_t = \text{acc}(\omega_t) - \text{acc}(\omega_{t-1})$. This formulation maximizes the model's performance while efficiently managing the computational resources and training latency.

Chapter 5. Preliminary Experimental Study

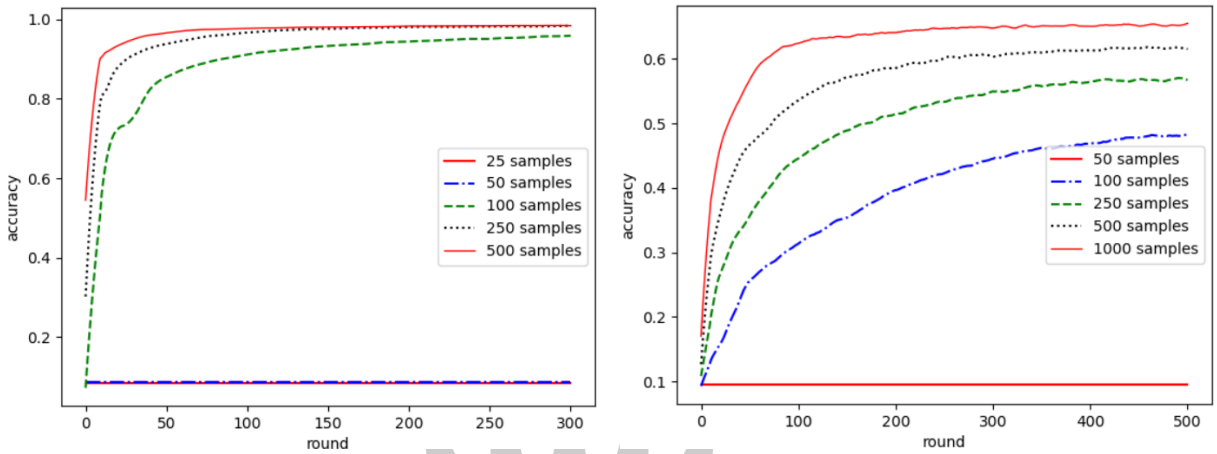
5.1 Impact of non-IID Features on Accuracy

Local datasets gather different behaviors and environments, making different data quality between the clients and motivating us to select the best set of clients to improve the performance and convergence speed. Before we solve the client selection problem, we conduct several experiments on different non-IID features of the dataset, including quantity-skew, label-skew level, and mislabel level. The aim is to provide insight into how these non-IID features affect the performance and convergence speed through the results of the experiments.

In the following experiment settings, unless otherwise specified, we employed the FedAvg algorithm with a consistent number of selected clients in each global training round, $N_T = 10$. The local iteration count is $I_{\text{loc}} = 10$, the batch size is $B_{\text{loc}} = 64$, the learning rate is $\gamma_{\text{loc}} = 0.005$, and there were 100 clients, each with an IID dataset with 200 samples. To reduce the variance of the result, each result was averaged over ten experiments

5.1.1 Quantity-Skew Impact

In centralized machine learning, it is well-established that larger datasets generally lead to higher performance and faster convergence speeds. To investigate whether this principle holds in a federated context, we conduct experiments on two distinct datasets: MNIST [87] and



(a) Accuracy evaluation with the MNIST dataset: To demonstrate the impact of dataset size, we evaluate five scenarios where clients assess local datasets of sizes 25, 50, 100, 250, and 500 samples, respectively, in different scenarios.

(b) Accuracy evaluation with the CIFAR-10 dataset: To demonstrate the impact of dataset size, we evaluate five scenarios where clients assess local datasets of sizes 50, 100, 250, 500, and 1000 samples, respectively, in different scenarios.

Figure 3: Quantity-skew impact

CIFAR-10 [88]. We implemented five scenarios for each dataset, each with a different dataset size (number of samples). In each scenario, all clients possessed uniform-sized dataset sizes.

The results Fig. 3 confirm that, as observed in previous studies and centralized learning mechanisms, larger client dataset sizes in both datasets correlate with improved test accuracy and faster convergence. Additionally, datasets below a certain threshold fail to support effective learning. Notably, in the more complex CIFAR10 task Fig. 3b, the performance disparities across different dataset sizes are even more pronounced.

5.1.2 Label-Skew Impact

Label skew refers to a type of non-IID data distribution characterized by differences in label distribution across clients. Clients with more uniform datasets or encompass all labels can contribute more comprehensively to the global model. This typically results in enhanced model performance. Conversely, when underrepresented labels dominate clients' distributions, the global model's performance may suffer due to insufficient information from these labels.

To understand how label skew affects training, we explored the following scenarios to simulate varying levels of label skew:

1) **IID**: Each client possesses an equal distribution of all ten labels, ensuring no label skew.

2) **SingleLabel**: Each client exclusively possesses samples from a single label, representing extreme label skew.

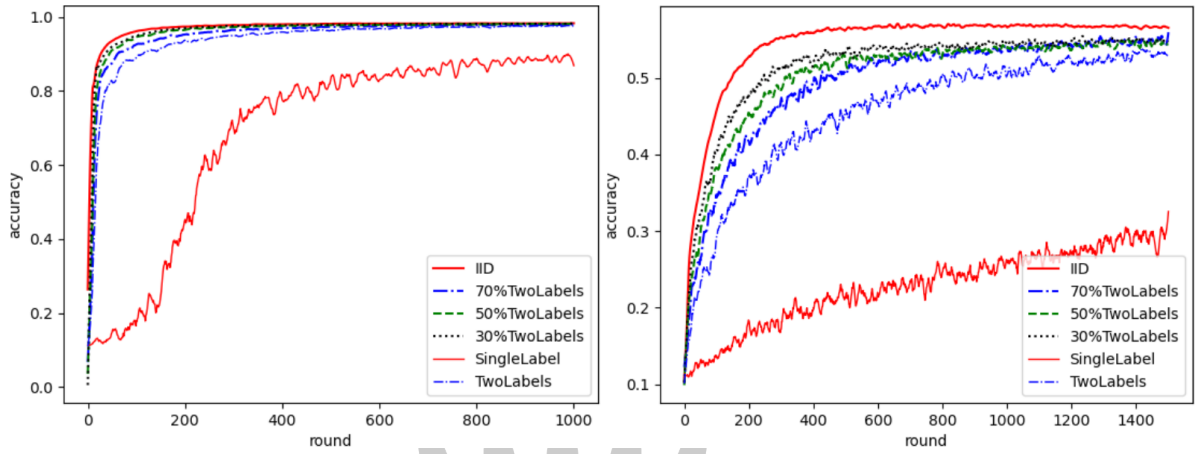
3) **TwoLabels**: Each client has samples from two labels, ensuring an equal sample count for both labels. With 90 clients, we ensured that no more than two clients shared the same label combination.

4) **70%TwoLabels**: A hybrid scenario where 30% of clients follow an IID setting, while the remaining 70% follow the nonIID-label2 setting.

5) **50%TwoLabels**: A balanced scenario with 50% of clients in IID settings and 50% in nonIID-label2 settings.

6) **30%TwoLabels**: A predominance of IID data with 70% of clients in IID settings and 30% following the nonIID-label2 setting.

The result Fig. 4b from the CIFAR-10 dataset reveals a clear trend that, as the level of label-skew increases, both test accuracy and convergence speed worsen. On the other hand, The result Fig. 4a from the MNIST dataset shows only slight performance declines under similar conditions, except in the extreme case of **SingleLabel**. This suggests that the severity of performance impact from label-skew depends on the complexity of the task, revealing that different tasks have varying levels of resilience to this issue.



(a) Accuracy evaluation with MNIST dataset Under FedAvg, each round t includes ten clients, each with a uniformly sized dataset of 200 samples

(b) Accuracy evaluation with CIFAR-10 dataset Under FedAvg, each round t includes ten clients, each with a uniformly sized dataset of 200 samples

Figure 4: label-skew impact

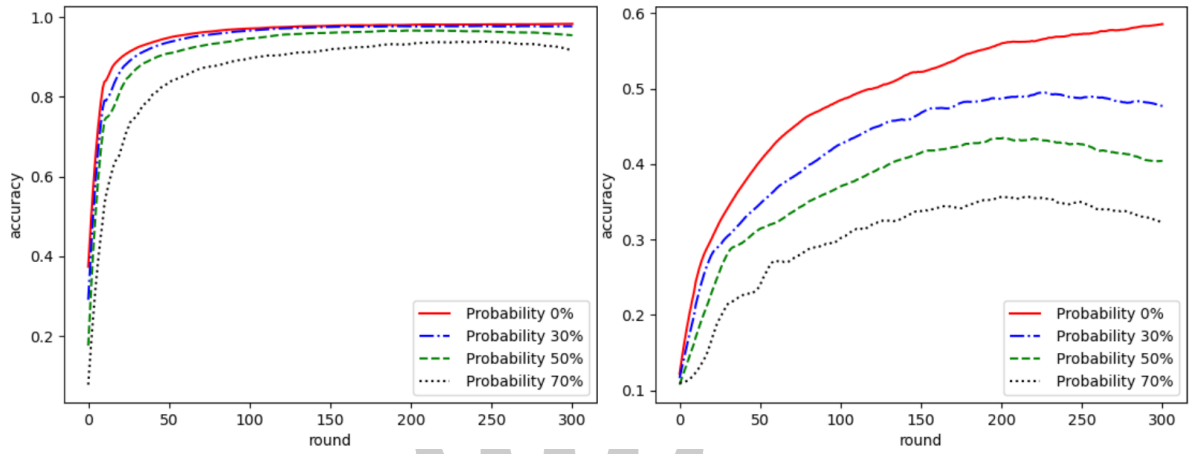
5.1.3 Mislabel Impact

Mislabeling occurs when a training sample is incorrectly labeled, potentially steering the global model in the wrong direction and resulting in diminished performance. To investigate how mislabeling degrades model performance, We explored three common mislabel scenarios mentioned in the literature:

1) **Random Mislabeling With Probability** y: In this scenario, each sample in each client's dataset has a probability of $y\%$ being mislabeled with a random label.

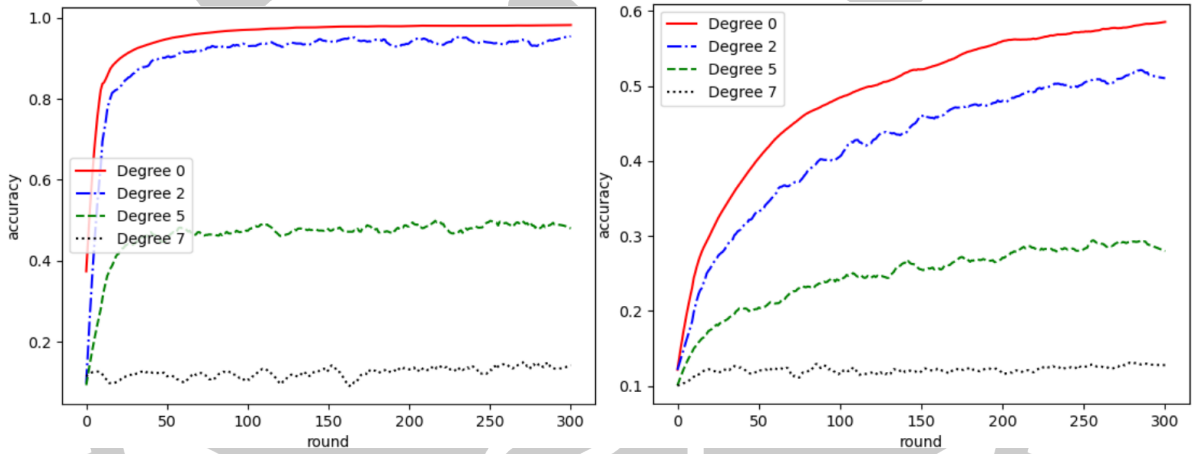
2) **Sequential Mislabeling with Degree** y: In this scenario, mislabel level y specifies the number of labels that are inaccurately labeled. For example, when $y = k$, all samples with labels $1, 2, ..., k$ are relabeled as labels $2, 3, ...k + 1$, respectively.

3) **Cyclic Mislabeling With Degree** y: In this scenario, 90 clients are involved, and y labels are mislabeled cyclically. It differs from Sequential Mislabeling in that all samples with label k are relabeled as label 1. For example, in **Cyclic Mislabeling With Degree** 3, three labels, 2, 5, and 7, are relabeled as 5, 7, and 2, respectively.



(a) MNIST (b) CIFAR-10

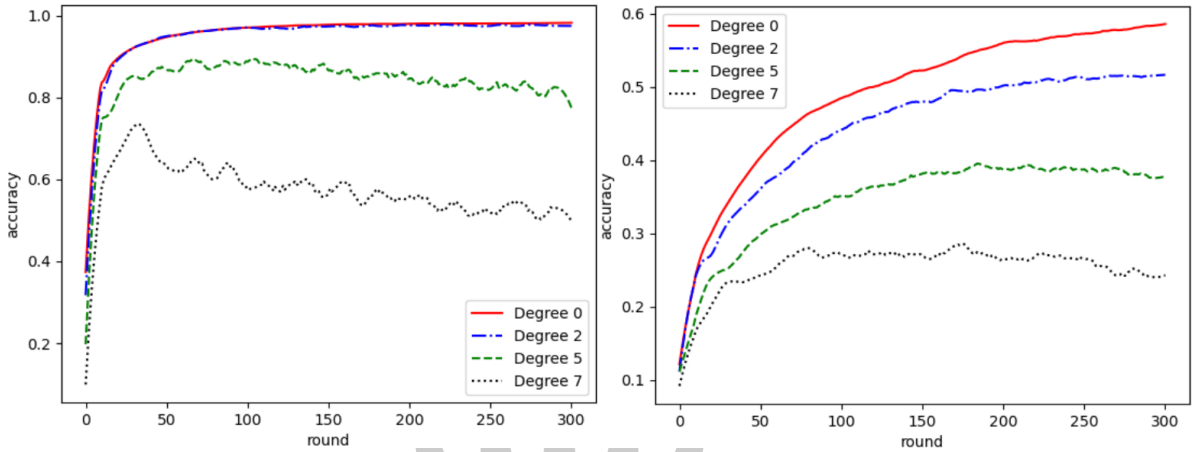
Figure 5: Random Mislabeling With Probability $y \in \{0\%, 30\%, 50\%, 70\%\}$



(a) MNIST (b) CIFAR-10

Figure 6: Sequential Mislabeling with Degree $y \in \{0, 2, 5, 7\}$

From result Fig. 5, Fig. 6 and Fig. 7 across three distinct scenarios and two datasets, we note a marked decline in performance correlating with increased degree of mislabeling. In scenarios involving label-skew features (refer to Fig. 4a and Fig. 4b), learning was achievable albeit at a reduced efficiency. However, the scenarios demonstrated a pronounced drop in performance. This was especially evident in sequential mislabeling with $y = 7$, where the learning process failed. This observation suggests that the selection mechanism should exclude clients suffering from severe mislabeling. Moreover, the impact of mislabeling was notably more severe in the CIFAR-10 dataset. This suggests that the complexity of the task exacerbates the challenges posed by inaccurate labels. Among the three mislabel scenarios, the sequential mislabel sce-



(a) MNIST (b) CIFAR-10

Figure 7: Cyclic Mislabeling With Degree $y \in \{0, 2, 5, 7\}$

nario shown in Fig. 6 consistently resulted in the most significant performance degradation, highlighting its potential as a stress test for further refining our approach, so we will focus on the sequential mislabel scenario in our upcoming experiments to better understand the dynamics of mislabeling and develop more robust federated learning systems.

5.1.4 Fairness Impact

Recent studies [18–20] show that feature-skew is a significant non-IID issue in federated learning, characterized by varying relationships between features and labels across different clients. For instance, in the MNIST dataset, the numeral three might be represented in various styles, while in the CIFAR-10 dataset, animals could be photographed from different angles. This results in unique feature sets for the same labels. Different from label-skew, which can be statistically analyzed, feature-skew poses a more significant challenge for quantification due to its ambiguous definitions and widespread occurrence.

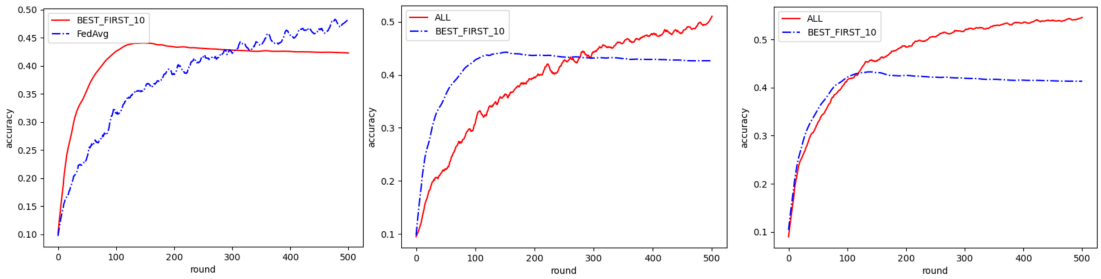
In light of this, we do not view feature-skew negatively but as an essential element of data diversity that ensures effective model generalization across diverse feature representations within the same label. Therefore, it is crucial to implement fairness in client selection. Such fair-

ness ensures that the model adapts to and benefits from the intrinsic variations within the data, enhancing its robustness and general applicability.

In the following sections, we will explore how fairness in client selection influences the model's performance and convergence speed. To understand this, we conducted experiments on two non-IID features: label-skew and mislabel.

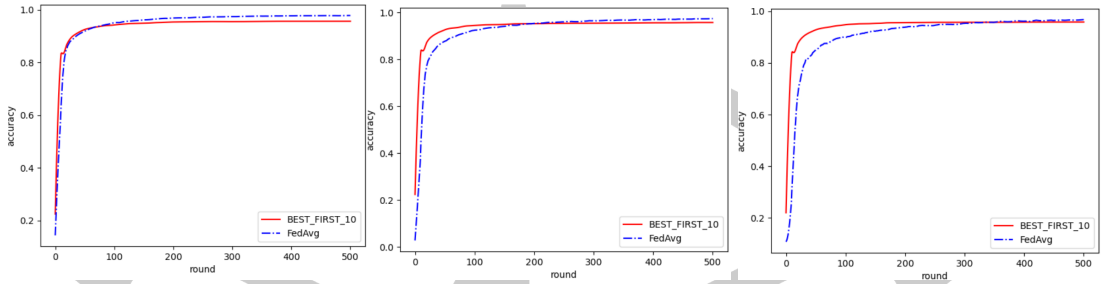
1) For **label-skew feature**, there are three non-IID distributions, **10%TwoLabels**, **10%FourLabels**, and **10%SevenLabels**. To understand the selection dynamics, we compared two client selection mechanisms. The first, **FedAvg**, presents the fairness-oriented selection mechanism. It randomly selects clients without considering the quality of their datasets, which results in nearly uniform selection frequencies across clients. The second, **IID_ONLY**, exclusively selects clients with IID datasets, demonstrating a lack of fairness due to excluding the non-IID clients.
2) For **mislabel feature**, we evaluated the scenarios **80% client with Sequential Mislabeling with Degree** 3 and **80% client with Sequential Mislabeling with Degree** 5. Under these conditions, **FedAvg** again presents the fairness-oriented selection mechanism. In contrast, the **CORRECT_ONLY** mechanism selects only those clients with accurately labeled data, potentially compromising fairness by excluding those with higher mislabeling rates.

Our experiments reveal different results from two non-IID features—label-skew and mislabel. For the label-skew feature shown in Figs. 9 and 8, we observe that fairness in client selection significantly improved accuracy. However, more global rounds were required to achieve convergence, and the importance of fairness increased as the level of label-skew decreased. However, in the case of the mislabeled feature shown in Figs. 10c and 10d, implementing a fairness selec-



(a) 10%TwoLabels (b) 10%FourLabels (c) 10%SevenLabels

Figure 8: fairness impact with label-skew feature, CIFAR10



(a) 10%TwoLabels (b) 10%FourLabels (c) 10%SevenLabels

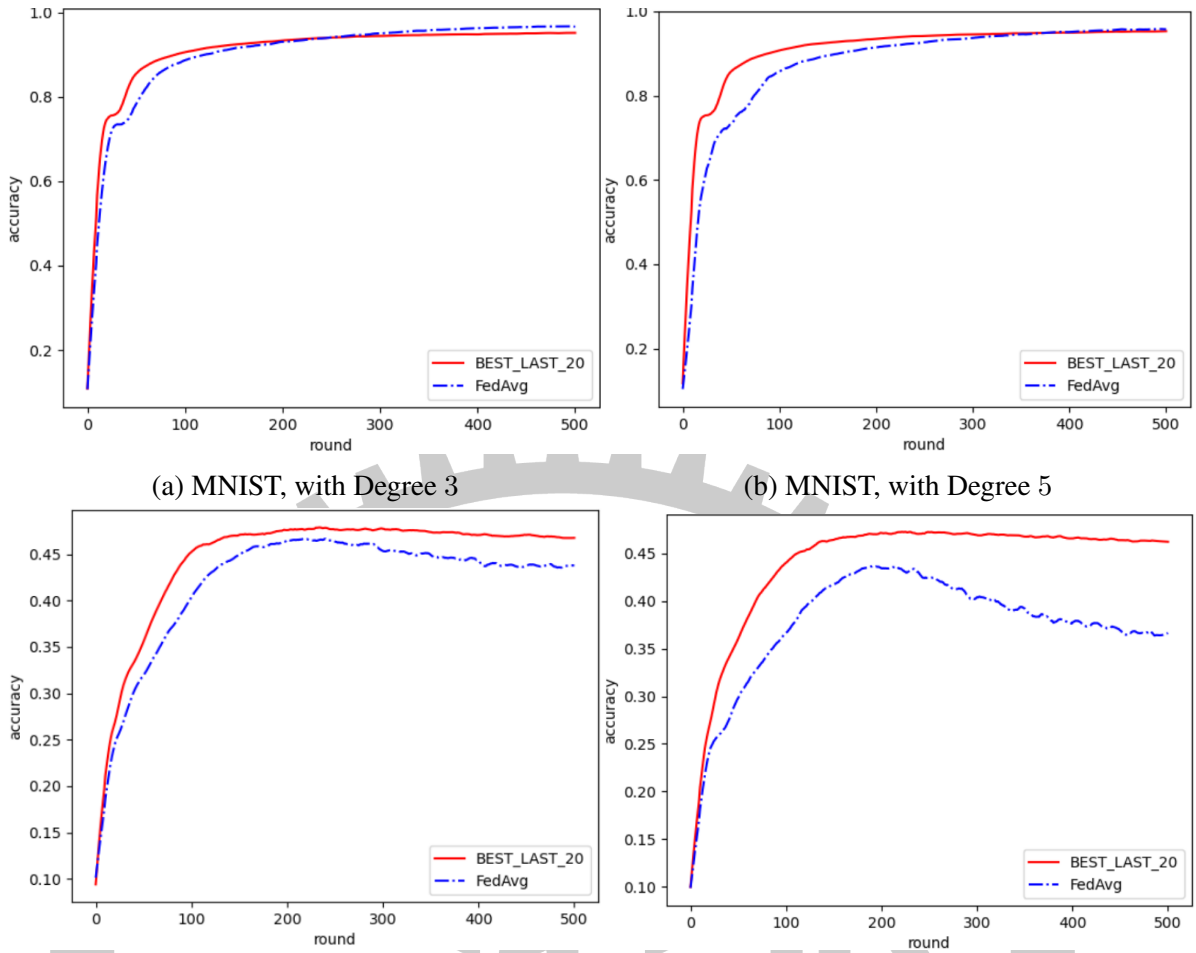
Figure 9: Fairness impact with label-skew feature, MNIST

tion mechanism significantly reduced performance, suggesting that mislabeled samples do not contribute valuable learning information.

It is particularly worth noting that in the MNIST dataset, where tasks are generally easier to learn, the impact of fairness mechanisms in mislabeled scenarios (Figs. 10a and 10b) and unfair mechanisms in label-skew scenarios (Fig. 9) showed similar performance and convergence rates compared to other mechanisms. This suggests that the complexity of the task influences the effectiveness of fairness-oriented approaches in federated learning environments.

5.2 How to Measure these factors?

Our experiments have demonstrated that dataset size, label skew, and mislabel levels significantly affect task performance and convergence speed in FL. Given the importance of privacy in FL, the server is strictly prohibited from accessing any distributional information from the local datasets, including label and feature distribution, except for dataset size. This restriction



(a) MNIST, with Degree 3

(b) MNIST, with Degree 5

(c) CIFAR10, with Degree 3

(d) CIFAR10, with Degree 5

Figure 10: Fairness impact with 80% with Sequential Mislabeling

calls for a robust evaluation mechanism that can accurately assess the quality of local datasets without breaching privacy. In the following, we introduce the method to evaluate the quality of the clients' dataset under privacy requirements and how to score datasize and quality.

5.2.1 Datasize Score

In typical FL studies, the server has access to the information about the dataset sizes of participating clients. To enhance our algorithm's design and ensure equitable client involvement, we define the data size score, d_i, as the normalized size of each client's dataset. This normalization process involves mapping each client's dataset size to a range from 0 to 1. The formula

for this normalization is as follows

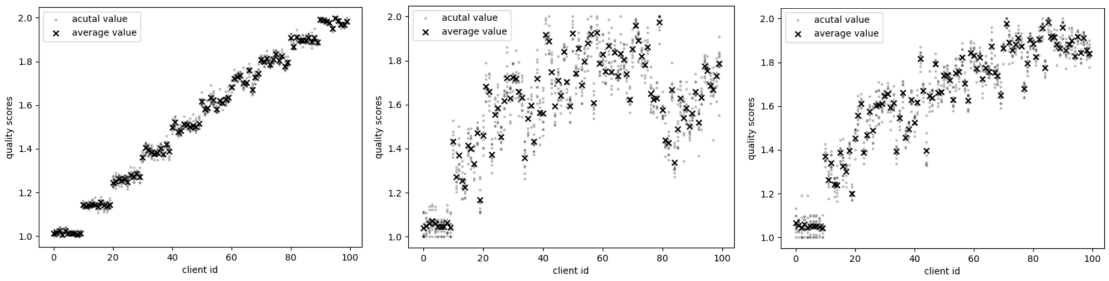
$$d_i = \frac{D_i - \min_{j \in C} D_j}{\max_{j \in C} D_j - \min_{j \in C} D_j}, \tag{5.1}$$

where D_i is the datasize of client i.

5.2.2 Quality Score

The term quality here reflects the joint impact of mislabeling and label-skew. The study [69] assesses mislabel levels by initially applying a feature extractor to the local dataset, followed by using a k-nearest Neighbor (kNN) approach to identify noisy samples. We use the ratio of noisy to correct-label samples as the client's mislabel level. However, due to concerns about the feasibility of this method and the risk of untrusted clients manipulating their reported accuracy to increase their selection chances, we adopted an alternative approach as suggested by [54]. In the methodology, to evaluate mislabel levels at the initial stage of FL, the server distributes an initial model to all clients, who then train this model on their local datasets for $\widetilde{I}$ local epochs. Then the server evaluates these uploaded local models using its test dataset, denoting the evaluated accuracy of client i as $\tilde{\text{acc}}_i$. The test accuracy of these models serves as an indicator of the clients' mislabeled levels. Furthermore, our methodology also proves effective in evaluating label-skew levels of clients. Therefore, we employ the same approach to assess mislabel and label-skew levels simultaneously. We denote the evaluated value as the quality score of client i, represented as q_i, which is calculated by the following formula

$$q_i = \frac{\tilde{\text{acc}}_i - \min_{j \in C} \tilde{\text{acc}}_j}{\max_{j \in C} \tilde{\text{acc}}_j - \min_{j \in C} \tilde{\text{acc}}_j} \tag{5.2}$$



(a) MNIST, 500 samples (b) CIFAR-10, 500 samples (c) CIFAR-10, 1000 samples

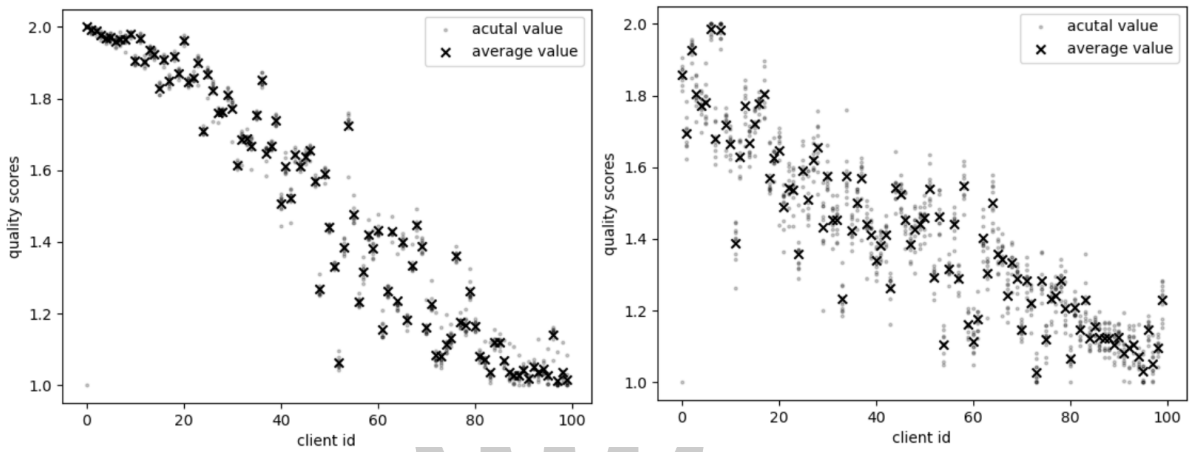
Figure 11: Quality score under the label-skew feature

Here, $\widetilde{I}$ represents the initial accuracy of client i's local training model after E_v epochs, evaluated against the global test dataset D_g. This scale ensures that the client with the lowest test accuracy scores 0, while the highest scores 1.

To validate the efficacy of our adopted method in evaluating both label-skew and mislabel levels, we conducted two experiments under the label-skew and mislabel features:

1) **Label-Skew Evaluation**: This experiment involved 100 clients, each with a dataset of 500 samples. To demonstrate various levels of label-skew, we configured the datasets so that the first ten clients had only one label. For every subsequent group of 10 clients, the number of unique labels increased by one. The local training epochs for evaluation E_v was set to 30.

   The results shown in Fig. 11 showed that the quality score increases with the client ID, where the clients with higher IDs possess more labels. However, there was a performance drop in the CIFAR-10 dataset for clients ranging from ID 70 to 80. This decline is because each client's total dataset size was fixed at 500 samples. Consequently, to maintain this size, the number of samples per label decreases, leading to insufficient data per label for effective learning. To validate this assumption, we conducted a follow-up experiment with the same settings but increased the number of samples per client to 1000. This adjustment showed an increasing trend in performance. We are confirming our hypothesis that a more



(a) MNIST (b) CIFAR-10

Figure 12: Quality score under the mislabel feature

significant number of samples per label enhances learning effectiveness.

2) **Mislabel Level Evaluation**: Similarly, this experiment also included 100 clients, each possessing a dataset of 500 samples. To explore different levels of mislabeling, the datasets for the first ten clients contained correctly labeled data. For every subsequent group of 10 clients, the mislabel rate increased by 10%. Like in the label-skew experiment, the local training epochs for evaluation E_v was set to 30. The trend shows that the quality score decreased as the client ID increased, whereas the clients with higher IDs suffered from more server mislabeling.

The results of both experiments validate the effectiveness of our method in accurately assessing the quality of each client's data. This accuracy enables the server to make more informed and strategic client selections.

5.2.3 Fairness Score

The concept of fairness mentioned earlier ensures that each client is selected an equal number of times and at equal frequencies to prevent local optimization. To illustrate how long clients have yet to be selected, we introduced the concept of a virtual queue, which each client

maintains. At the beginning, each client's virtual queues are set to 0. In each round, the virtual queue of each client increases by β, called the fairness rate, which ranges from 0 to 1. If a client is selected in round t, its virtual queue decreases by 1. Therefore, the virtual queue or fairness score of client i at round t, denoted as f_i^t, can be represented as

$$f_i^t = \begin{cases} 0 & t = 0, \\ f_i^{t-1} + \beta - a_i^{t-1} & \text{otherwise}, \end{cases} \tag{5.3}$$

where a_i^{t-1} is an indicator that shows whether client i was selected in round $t-1$. A higher fairness score for a client implies that the client has not been selected for longer since it last participated in the training, indicating that we should prioritize selecting this client for fairness. This formula identifies how long a client has not been selected since the last time and ensures fairness by always selecting the client with the highest fairness score.

However, in our study, we also consider battery constraints. Initially, we can ensure that each client has the same impact on the global model because each client has enough battery to participate in training with a similar frequency. As the process continues, some clients may run out of battery and cannot be selected anymore, which can lead to the global model becoming locally optimized. A better strategy is reducing the participation frequency of clients with lower battery capacity. This approach ensures that these clients are selected only sometimes in the initial stages while still maintaining their impact in the later stages. To achieve different participation frequencies for each client, we adjust the fairness rate in the fairness score based on each client's battery capacity or participation times. The adjusted fairness rate,β_i^t, is calculated by dividing the remaining battery after round $t-1$ by the predicted energy consumption $\hat{e}_i^t$, and it ranges from 0 to N_T. We can formulate β_i^t as

$$\beta_i^t = \frac{N_T \varsigma_i^t}{\sum_{j \in C_a} \varsigma_j^t}, \tag{5.4}$$

where $\varsigma_i^t = \frac{b_i^t}{\hat{e}_i^t}$ is the potential participation time, and β_i^t is the normalized ς_i^t multiplied by N_T, which ensures the stability of the queue. We reformulate the clients' fairness score as

$$f_i^t = \begin{cases} 0 & t = 0, \\ f_i^{t-1} + \beta_i^t - a_i^{t-1} & \text{otherwise.} \end{cases} \tag{5.5}$$

Chapter 6. Algorithm

To solve problem P2, we first explore preliminary studies that qualify the quality of clients' datasets and discuss their pros and cons. Then, we propose a Shapley value-based approach that predicts each client's accuracy contribution and solves the optimization problem P2 using this prediction along with the clients' latency for the round.

Our approach comprises four main components. The first part, detailed in 6.2, involves the GTG-Shapley value [89], which is used to evaluate the clients' contributions in each round efficiently. The second part, described in Section 6.3, adopts a ridge regression model, benefiting from its ability to mitigate the gap in contribution due to high variance and insufficient selection times of clients; we then use its predictions as the clients' predicted contributions. The third part section, in Section 6.4, introduces a fairness-aware design that ensures online client selection considers fairness at a global level under different non-IID distributions. Lastly, in Section 6.5, we solve the optimization problem P2 using traditional optimization methods.

6.1 Preliminary Evaluation Methods for Client Data Quality

A primary challenge in developing a client selection mechanism to achieve the optimal balance between latency and accuracy progress is that the accuracy progress remains unknown before the global training round is completed. This inherent uncertainty complicates using traditional optimization algorithms to identify the best trade-off. Consequently, there is a pro-

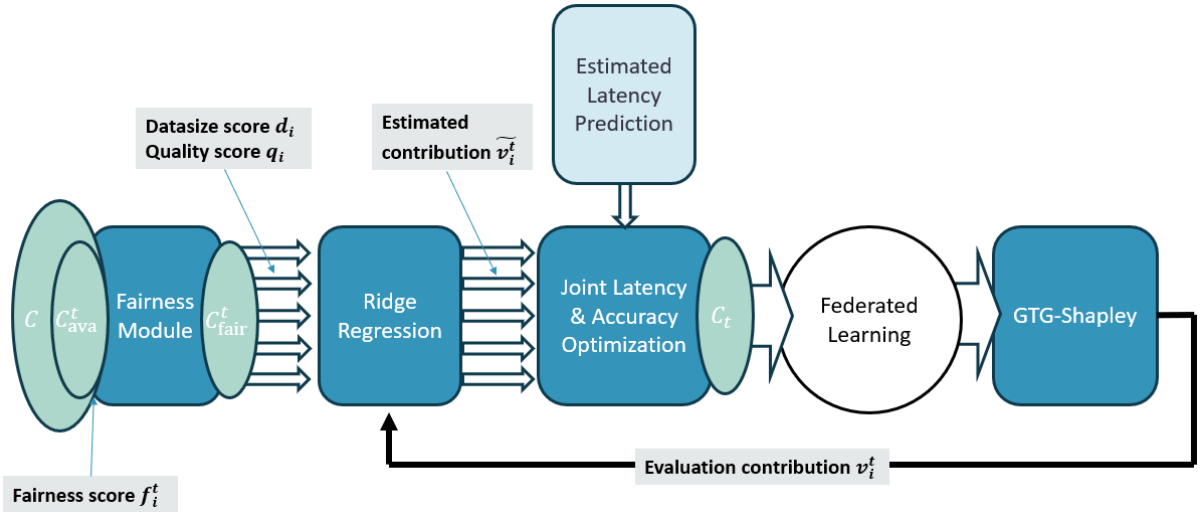


Figure 13: Enter Caption

nounced need for an efficient method to predict client contributions. To tackle this issue, we explore several methodologies to evaluate client contributions, each designed to mitigate the uncertainties and enhance the effectiveness of the selection process.

1) **Optimization** or **Game Theory-Based Methods**: These approaches model the client's contribution using factors such as dataset size, evaluated metrics, or historical performance. However, finding the most accurate model to describe the relationship between a client's estimated contribution and these factors is a significant challenge. The complexity of defining a universally applicable model makes this approach particularly challenging.

2) **Multi-Armed Bandit (MAB)** and **Reinforcement Learning (RL) Based Methods**: The MAB-based methods utilize feedback from global model performance to predict individual contributions. MAB theory assumes that each client has a "fixed" contribution level. However, experimental observations suggest that actual contributions vary at different stages of the learning process. This variability necessitates periodic selection and re-evaluation of each client to gauge their contributions accurately, making it challenging to maintain a static selection strategy. RL-based methods employ neural network models to enhance the precision of contribution estimations. They require extensive pre-training on

FL tasks to develop the neural network model. Like MAB, RL methods struggle with dynamically updating the prediction model across different stages of FL due to the inherent complexities of updating neural network models.

3) **Shapley Value (SV)**: The Shapley Value (SV), originating from Cooperative Game Theory, offers a robust framework for quantifying individual contributions and ensuring fairness within a group. The study [89] evaluates client contributions in FL using three different methods: the Influence Mechanism, the Reputation Mechanism, and the Shapley Value across varying levels of data quality. The findings highlight that the Shapley Value is exceptionally effective at discerning the quality of client datasets and equitably distributing contributions among clients. In the following, we adopt the Shapley Value-based method to predict the contribution of each client.

6.2 Efficient Shapley Value

In FL, the Shapley value is defined as the client's contribution to the accuracy progress. To evaluate it, the utility of a subset S can be defined as the accuracy progress after aggregating the updated local model of the clients in subset S, formulated as

$$U(S, \omega) = \text{acc}(A(S, \omega)) - \text{acc}(\omega), \tag{6.1}$$

where $\text{acc}(\omega)$ is the test accuracy of the global model ω and the $A(S, \omega)$is the aggregation function that returns the updated model with the training participants in S. Then, the Shapley

value of client i at the round t is

$$s_i^t = \begin{cases} \sum_{C' \subseteq C_{\text{suc}}^t \setminus i} \frac{|C'|!(|C_{\text{suc}}^t|-|C'|-1)!}{|C_{\text{suc}}^t|!} (U(C' \bigcup \{i\}, \omega^t) - U(C', \omega^t)) & , i \in C_{\text{suc}}^t \\ 0 & , i \notin C_{\text{suc}}^t \end{cases}. \tag{6.2}$$

If a client does not belong to the set of successfully updated clients or did not participate in the round, its Shapley value is set to 0.

However, despite the proven effectiveness of Shapley value-based approaches in evaluating client contributions, several challenges complicate their application to our client selection process

1) **Exponential time complexity**: To comprehensively and fairly assess client contributions, the Shapley value method evaluates the marginal contributions of each client under all possible combinations of selected client subsets within a round. This approach, while thorough, requires calculations that grow exponentially with the number of clients, posing significant computational challenges. The exhaustive nature of these calculations ensures fairness and accuracy in contribution assessment but can be impractical for large-scale FL environments due to the intense computational demands.

2) **High Variance in Shapley value Online**: Our experiments have shown that there is significant variance in the Shapley value measurements for the same client across different global rounds. The variance is caused by variations in the combination of selected client sets and the evolving state of the task model. Traditional Shapley value-based methods focus on contributions over the entire duration of FL training, with evaluations typically conducted after training concludes. This approach allows for the mitigation of variance through sufficient selection occurrences. However, for real-time determinations,

this leads to substantial discrepancies in measurement.

3) **Unbalanced Selection**: Previous studies using the Shapley value often employ a random selection strategy, which ensures each client is selected with similar frequency and provides adequate opportunities for participation. However, in our study, we aim to enhance model performance and convergence speed by selecting clients based on variations in data quality and hardware resources. This approach can result in some clients being selected less frequently due to initially lower measured Shapley values, even though they may be. Consequently, we may miss out high-contribution clients.

To overcome the challenge of evaluating the Shapley value with exponential time complexity, the study [89] introduced a time-efficient version of the Shapley value evaluation method tailored for FL, known as the GTG-Shapley value. This innovative approach ensures results comparable to those obtained from the complete Shapley value evaluation while reducing the computational time to $O(N \log N)$.

The GTG-Shapley value method implements two key improvements. First, it utilizes the Monte Carlo estimation technique to approximate Shapley values. Instead of calculating the marginal utility for all possible permutations within the subset, this method randomly samples permutations and calculates the average marginal contribution of client i. Then, we use the average values of the i in each calculated sampling permutation as the predicted Shapley value of i. This sampling continues until the estimated Shapley value stabilizes, eliminating the need to exhaustively compute the marginal utility for every possible permutation. The approximate Shapley value is defined as

$$s_i^t = \mathbb{E}_{\pi \simeq \Pi}[U(C_\pi^i \bigcup \{i\}, \omega^t) - U(C_\pi^i, \omega^t)], \tag{6.3}$$

where Π is the set of all the possible permutations in C^t_{suc} and C^i_π is the number i element in the permutation π.

Secondly, observations from FL suggest that the marginal contribution of clients typically diminishes as their order in the permutation increases. The most significant contributions usually occur early in the sequence, while contributions from later positions are minimal and often negligible. This pattern facilitates a strategic reduction in computations by establishing a threshold for marginal contributions. Calculations can be prematurely terminated once contributions drop below this threshold during a permutation analysis. This method significantly reduces the number of operations required for each sampled permutation, further improving the efficiency of Shapley value approximations.

Although there is an efficient method for calculating the Shapley values proposed for FL, Shapley values only reflect comparative differences and do not indicate the absolute contributions of each client. This scaling issue introduces errors when performing the trade-off between accuracy progress and the latency of a round in problem P2. Therefore, rescaling the Shapley values to reflect the actual accuracy progress is essential.

To align Shapley values with the actual scale of accuracy progress observed in a given round, we rescale the Shapley value by multiplying the evaluated Shapley value s^t_i with a rescale coefficient r_t, thereby calculating a more representative contribution value v^t_i by

$$v^t_i = r^t s^t_i. \tag{6.4}$$

In the following, we consider two mechanisms to determine the rescale coefficient r_t, ensuring they accurately represent the tangible contributions and performance progress.

1) **Rescale Mechanism 1**: One intuitive approach to rescale the Shapley values by ensuring the sum of the contribution value matches the actual performance improvement achieved

Algorithm 1 RescaleMechanism1

Input: selected clients C, Shapley values of selected clients S and the accuracy progress Δacc

Output: evaluated contribution $[v_1, ..., v_{|C|}]$

$r \leftarrow \frac{\Delta\text{acc}}{\sum_{i \in C} s_i}$

$v_i \leftarrow r s_i, \forall i \in C$

return $[v_1, ..., v_{|C|}]$

in that round. Then, the coefficient r_t is calculated as

$$r^t = \frac{\Delta\text{acc}_t}{\sum_{j \in C^t_{\text{suc}}} s_j}. \tag{6.5}$$

This rescaling ensures that v_i^t is aligned with the overall performance improvement, thus providing a fair and proportional evaluation of each client's input based on the observed enhancements.

2) **Rescale Mechanism 2**: The previous calculation for Shapley values does not effectively represent the state of accuracy progress. Even when there is a trend of declining accuracy, several clients' contributions can be calculated as positive. To more accurately represent the trend of improvement, we propose a second method that considers the trend of actual accuracy progress when determining each client's contribution. If the accuracy progress is positive, the rescaling coefficient r_t is calculated by

$$r^t = \frac{\Delta\text{acc}(s_i - \min_{j \in C_t} s_j)}{\sum_{j \in C} s_j - \min_{j \in C_t} s_j}. \tag{6.6}$$

Conversely, if the accuracy progress is negative, r_t is calculated by

$$r^t = \frac{\Delta\text{acc}(s_i - \max_{j \in C_t} s_j)}{\sum_{j \in C} s_j - \max_{j \in C_t} s_j}. \tag{6.7}$$

Algorithm 2 RescaleMechanism2

Input: selected clients C, Shapley values of selected clients S and the accuracy progress Δacc
Output: evaluated contribution $[v_1, ..., v_{|C|}]$

if $\Delta\text{acc} > 0$ **then**
    $m \leftarrow \min_{i \in C} s_i$
    $r_i \leftarrow \frac{\Delta\text{acc}(s_i - m)}{\sum_{j \in C} s_j - m}, \forall i \in C$
else if $\Delta\text{acc} < 0$ **then**
    $M \leftarrow \max_{i \in C} s_i$
    $r_i \leftarrow \frac{\Delta\text{acc}(s_i - M)}{\sum_{j \in C} s_j - M}, \forall i \in C$
else
    $r_i \leftarrow 0, \forall i \in C$
end if
$v_i \leftarrow r_i s_i, \forall i \in C$
return $[v_1, ..., v_{|C|}]$

It ensures that all clients' contributions are non-negative if the accuracy progress in that round is positive and non-positive otherwise. This method provides a more aligned and fair assessment of contributions based on the actual direction of accuracy changes.

Finally, the contribution of each client i at the round t, v_i^t can be determined by Algorithm 3. Lines 1 to 18 detail the efficient computation of Shapley values. Once we obtain approximate Shapley values, the previously mentioned rescaling mechanism is applied to compute the contribution of each client, denoted as V.

6.3 Embracing All Factors

As previously noted, predicting client contributions in FL presents substantial challenges due to imbalanced selection frequencies and limited client selection opportunities. If we use the accumulated history evaluated contribution as the predicted contribution of the client directly, these issues can lead to inaccurate predictions of client contributions.

In the earlier sections, we explored how different non-IID features influence performance

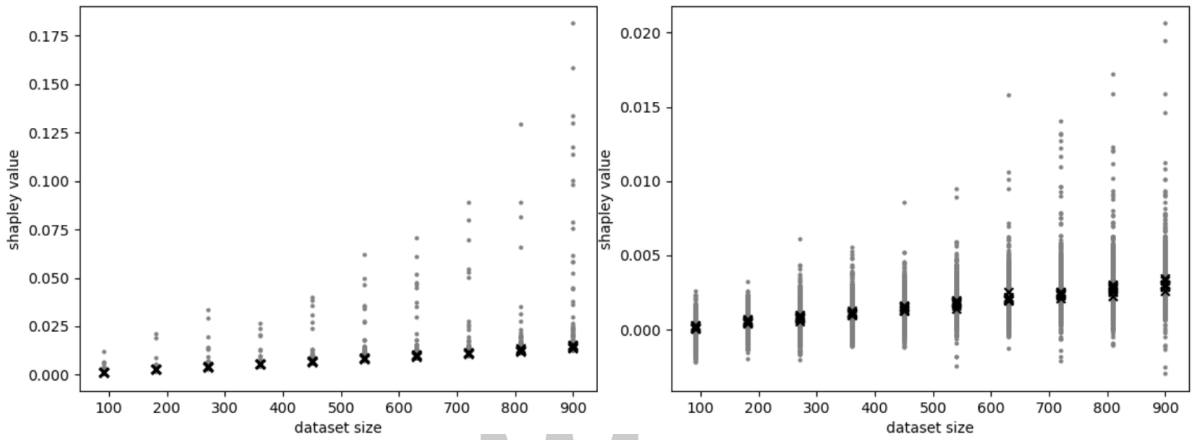
Algorithm 3 Shapley value

Input: utility function $U()$, model weight before aggregation ω_{t-1} and selected client set C_t
Output: evaluated contribution V

$k \leftarrow 0, \forall i \in C$
$s_i \leftarrow 0, \forall i \in C$
$\text{acc}_N \leftarrow U(\omega_{t-1}, C_t)$
if $|\text{acc}_N - 0| > \varepsilon_1$ **then**
  while *Convergence criteria* not met **do**
    sample the permutation π from Pi
    $v_{\text{pre}} \leftarrow 0$
    $v_{\text{now}} \leftarrow 0$
    for $j = 1...n$ **do**
      if $|\text{acc}_N - v_{\text{now}}| \geq \varepsilon_2$ **then**
        $v_{\text{pre}} \leftarrow v_{\text{now}}$
        $v_{\text{now}} \leftarrow U(\omega_{t-1}, C_\pi^j)$
      end if
      $s_{[C_\pi^j]} \leftarrow \frac{k-1}{k} s_{[C_\pi^j]} + \frac{1}{k}(v_{\text{now}} - v_{\text{pre}})$ # $[C_\pi^j]$ is the ID of j th client in π
    end for
    $k \leftarrow k + 1$
  end while
end if
$V \leftarrow$ ***RescalMechanism***$(S,$ ***acc***$_N)$
return V

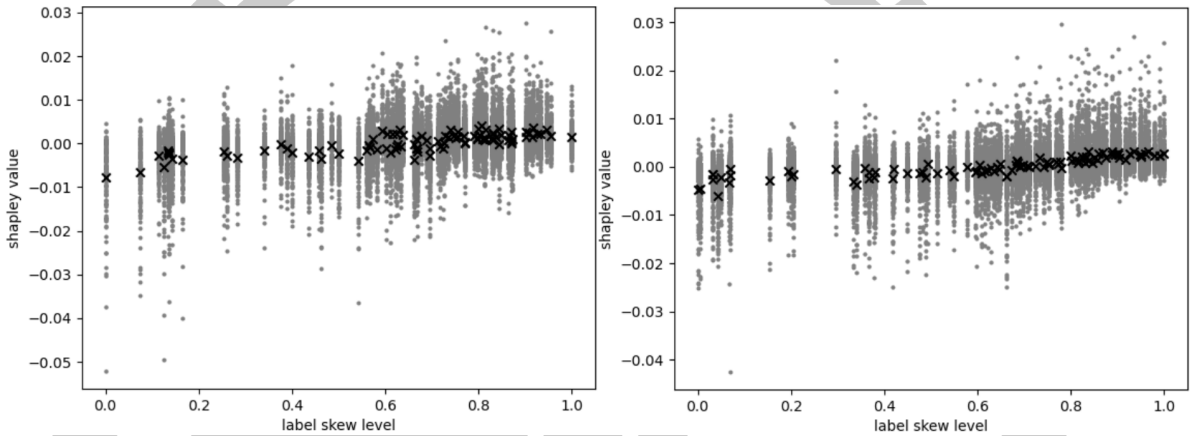
and convergence rates in FL systems. Building on this analysis, we then developed metrics to quantify the level of non-IID features. Using this foundation, we propose a predictive modeling approach that leverages pre-known information (e.g., quality score, datasize score, and fairness score) and historically evaluated contributions to more accurately predict client contributions. Our model incorporates non-IID factors as inputs to predict contributions, utilizing historically evaluated contributions measurements for training. The primary objective of this model is to ensure a fair and accurate assessment of all clients, especially those with fewer opportunities for selection.

To explore the viability of this method, we conducted experiments to examine the relationship between non-IID factors and the evaluated contribution of clients in an FL environment using the FedAvg algorithm with 100 clients. We conducted the experiments on three different



(a) MNIST (b) CIFAR-10

Figure 14: Dataset Size Impact on Shapley Value



(a) MNIST (b) CIFAR-10

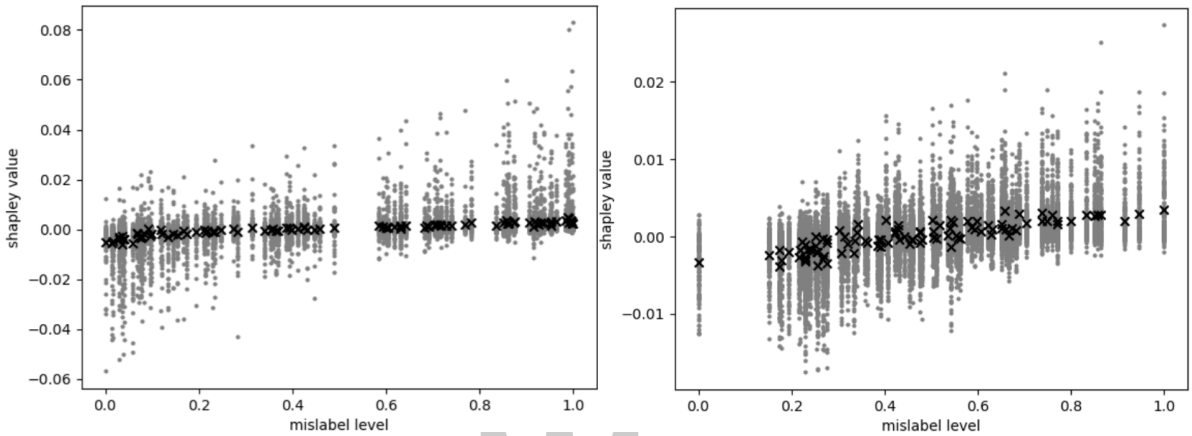
Figure 15: Label-Skew Level Impact on Shapley Value

non-IID features:

1) **Dataset Size Impact on Shapley Value**: In the first scenario, each of the initial ten clients had 100 samples in their local dataset, with each subsequent group of 10 clients receiving an additional 100 samples.

2) **Label-Skew Level Impact on Shapley Value**: In the second scenario, all clients started with 500 samples. The first ten clients had datasets containing only one label, and each subsequent group of 10 clients had one additional label.

3) **Mislabel Level Impact on Shapley Value**: Each client in the third scenario also had 500



(a) MNIST (b) CIFAR-10

Figure 16: Mislabel Level Impact on Shapley Value

samples. The first ten clients had correctly labeled datasets, with a 10% increase in the mislabel rate for each subsequent group of 10 clients, applied randomly.

The results show that, in all scenarios, the non-IID factor had a nearly linear relationship with the Shapley value. However, the impact of non-IID features on the Shapley value varied in magnitude.

Ridge regression is an enhanced linear regression that introduces a regularization component to prevent overfitting and effectively handles multicollinearity among predictors. Unlike traditional linear regression, ridge regression adjusts the objective function by adding an L2 regularization term. This adjustment aims to distribute the coefficients' magnitudes more evenly, thus improving the model's stability and its ability to generalize when predictors are highly correlated. The mathematical formulation of the objective function in ridge regression is

$$\min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda\|\psi\|^2. \tag{6.8}$$

Here, $\mathbf{Y}$ is the vector of observed historical target values with dimension $n \times 1$, where n represents the number of historical data points. $\mathbf{X}$ is the matrix of historical input features with dimension $n \times d$, where each feature vector x has a dimension of $1 \times d$ and d is the

number of feature dimensions. In this study, d equals 3, representing the quality score, datasize score, and a constant term, respectively. ψ is the vector of coefficients determined through the regularization process. λ is the regularization parameter that controls the strength of the penalty, which prevents overfitting and ensures a more robust model. The solution for the coefficient vector ψ in ridge regression can be derived as follows

$$\psi = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{6.9}$$

Experimental observations indicate that client contributions varied during different stages of the FL process. Early in the learning process, these contributions were more impactful than in later stages as the task neared convergence. This observation suggests that models should not weigh all historical data equally but prioritize recent interactions to reflect current conditions better. To implement this, we have incorporated a sliding window mechanism in our model, which maintains only the most recent data points within a predefined window length, n_{window}. This method removes outdated data pairs (x, y), keeping the model's predictions relevant and aligned with the latest data. This focus on recent contributions enhances prediction accuracy.

Our application continuously adds the learning results and selection information for each round. The computation of ψ involving $n \times d$ matrix multiplication is costly. Below, we introduce a ridge regression algorithm that incorporates new historical record pairs (x, y) and calculates ψ incrementally

1) **initialization**: The server maintains two matrices, A and B, with sizes $d \times d$ and $d \times 1$

respectively, initialized as

$$A = \lambda I_{d\times d} \tag{6.10}$$

$$B = 0_{d\times 1} \tag{6.11}$$

to manage data relevance within the window of interest, the queue Q with a maximum length of n_{window} is maintained to hold the recent data pairs.

2) **adding new history**: Each time a new history pair (x, y) is added, matrices A and B are updated to incorporate the new data by

$$A \leftarrow A + x^T \times x \tag{6.12}$$

$$B \leftarrow B + x^T \times y. \tag{6.13}$$

Then the new data pair (x, y) is added to the queue. If the queue's size surpasses the designated window limit n_{window}, it necessitates the removal of the oldest data pair $(x_{\text{old}}, y_{\text{old}})$ to maintain the queue length. Consequently, the contributions of the removed data are subtracted from matrices A and B by

$$A \leftarrow A - x_{\text{old}}^T \times x_{\text{old}} \tag{6.14}$$

$$B \leftarrow B - x_{\text{old}}^T \times y_{\text{old}}, \tag{6.15}$$

3) **prediction**: To predict the target value $\hat{y}$ for a new input x, we first calculate ψ:

$$\psi = A^{-1}B. \tag{6.16}$$

Algorithm 4 Client contribution predictor

Initialization(n_{window})

$A \leftarrow \lambda I_{d\times d}$

$B \leftarrow 0_{d\times 1}$

queue Q

Adding history $(x_{1\times d}, y_{1\times 1})$

if length of Q is equal to n_{window} **then**

$(x_{\text{old}}, y_{\text{old}}) \leftarrow Q$.pop()

$A \leftarrow A - x_{\text{old}}^T \times x_{\text{old}}$

$B \leftarrow B - x_{\text{old}}^T \times y_{\text{old}}$

end if

Q.pop(x, y)

$A \leftarrow A + x^T \times x$

$B \leftarrow B + x^T \times y$

prediction $x_{1\times d}$

$\psi \leftarrow A^{-1}B$

$\hat{y} \leftarrow x\psi$

return $\hat{y}$

Then, we compute the predicted value $\hat{y}$ by

$$\hat{y} = x\psi. \tag{6.17}$$

Now, there is a simple learning and adaptive client contribution prediction algorithm. We can treat the client's prediction contribution as the accuracy progress the client may bring. An optimization-based method is provided to find the optimized client selection for the trade-off between latency and accuracy.

6.4 Fairness Design

Our previous experiments have shown that the importance of fairness varies depending on the level of non-IID (non-independent and identically distributed) data among clients. When there is a significant variance in the non-IID level, regardless of the specific feature, the im-

portance of fairness decreases. This is because high variance indicates the presence of both outstanding and terrible clients in the potential client set. Overemphasizing fairness may lead to selecting low-quality or low-contribution clients, especially those with mislabeled data. Conversely, when the variance in the non-IID level is low, fairness becomes the dominant factor impacting both convergence speed and model performance.

From these observations, we designed an adaptive fairness mechanism that adjusts the importance of fairness based on specific circumstances. In each round, instead of selecting clients directly from the potential set C^t_{ava}, which may result in only high-quality clients being selected, we select clients from a fairness-aware potential set C^t_{fair}. This set includes the top L^t_{fair} clients with the highest fairness scores in C^t_{ava}. The value of L^t_{fair} is determined by:

$$L_{\text{fair}} = \max(N_T, |C^t_{\text{ava}}|[\alpha_1 \frac{\sum_{i \in C^t_{\text{ava}}} (q_i - \overline{q})^2}{|C^t_{\text{ava}}|} + \alpha_2]), \tag{6.18}$$

where q_i is the quality score of client i and $\overline{q}$ is the mean quality score of C^t_{ava}. The parameters α_1 and α_2 are hyperparameters.

In this design, when the variance of the quality score is small, the length of L^t_{fair} is small, including few clients in C^t_{fair} with high fairness scores. In this case, fairness is dominant because the selected clients always have high fairness scores due to the small fairness-awareness set. However, when L^t_{fair} is large, C^t_{fair} is large, and there is a possibility of unfair selection because more high-quality clients with better datasets are included in the set, even though they may have lower fairness scores.

6.5 Joint Latency and Accuracy Optimization

Problem P2 requires maximizing both latency and accuracy progress simultaneously, with weights appropriately assigned. Despite its straightforward presentation, a significant challenge arises due to the unknown accuracy progress once a training round is completed. We address this by using a predictive model for client contributions introduced in the previous section. This model uses client-specific data and historical performance to predict clients' contributions, which are then aggregated to predict round accuracy, thus allowing us to tackle P2 within linear time constraints.

For each round, the server gets the latency of each client l_i^t and the predicted contribution of each client $\hat{v}_i^t$ from its ridge regression model. The clients are then sorted in ascending order based on their latencies. Starting with the first client and proceeding to the last, the server identifies an optimal subset C_{temp}^i for each client i. This subset consists of client i and those preceding it, selected to maximize the sum of their predicted contributions. The server then compares the total value of $\sum_{j \in C_{temp}^i} \hat{v}_j^t - \delta l_i^t$ with the current best value. The process continues until the server has evaluated the client with the highest latency.

The algorithm processes $|C|$ iterations, each requiring the identification of the optimal client subset that maximizes utility. We can achieve efficient management of this process by using a heap, allowing for constant-time $O(1)$ access to the minimum elements and logarithmic-time $O(\log(n))$ for insertions and deletions. Consequently, the overall time complexity for solving Problem P2 is $O(\sum_{i \leq |C|} i \log(N_T))$, which simplifies to $O(|C|^2 \log(N_T))$.

Algorithm 5 client selection mechanism

Input: Potential clients set C, latency of each client l_i^t, predicted contribution of each client $\hat{v}_i^t$, the selected number N_T
Output: the optimal value p_{opt} and the clients set C_{opt}

Sort C by the latency to $L[i]$
$(p_{\text{opt}}, C_{\text{opt}}) \leftarrow (0, \emptyset)$
for $i = N_T ... n$ **do**
  $(p_{\text{temp}}, C_{\text{temp}}) \leftarrow (0, \emptyset)$
  for $j = 1 ... i-1$ **do**
    if $|C_{\text{temp}}| < N_T - 1$ **then**
      $p_{\text{temp}} \leftarrow p_{\text{temp}} + \hat{v}_{L[i]}^t - l_i^t$
      $C_{\text{temp}} \leftarrow C_{\text{temp}} \bigcup \{L[i]\}$
    else if $\min_{k \in C_{\text{temp}}} \hat{v}_k^t < \hat{v}_{L[i]}^t$ **then**
      $m \leftarrow \text{argmin}_{k \in C_{\text{temp}}}$
      $p_{\text{temp}} \leftarrow p_{\text{temp}} - \hat{v}_m^t + \hat{v}_{L[i]}^t$
      $C_{\text{temp}} \leftarrow C_{\text{temp}} \bigcup \{L[i]\} \setminus \{m\}$
    end if
  end for
  if $p_{\text{opt}} < p_{\text{temp}}$ **then**
    $(p_{\text{opt}}, C_{\text{opt}}) \leftarrow (p_{\text{temp}}, C_{\text{temp}})$
  end if
end for
return $p_{\text{opt}}, C_{\text{opt}}$

Chapter 7. Performance Evaluation

In this chapter, we conduct several experiments to demonstrate the efficiency of our solution.

7.1 Simulation Setting

We conducted two tasks on different image classification datasets, MNIST and CIFAR-10, using the neural network structure mentioned in FedAvg. In the MNIST dataset, each sample has a size of $28 \times 28 \times 1$ bytes, while in CIFAR-10, each sample has a size of $64 \times 64 \times 3$ bytes. The model weight size $|\omega_i^t|$, is 2.3MB for MNIST and 3.4MB for CIFAR-10. The dataset size for each client is equal to the sample size multiplied by the number of samples the client possesses. The hyperparameters for the experiments, the Global rounds I_g was 300 for MNIST and 400 for CIFAR-10. Batch size B_{loc} was 64 for both MNIST and CIFAR-10. Learning rate γ_{loc} for client training was 0.005 for both MNIST and CIFAR-10. Local iterations I_{loc} was 10 for both MNIST and CIFAR-10. We list the setting in Table 2.

For hardware heterogeneity, the transmission rate of each client was randomly generated between 100 Mbps and 10 Gbps, based on the 5G specifications from ITU-R M.2083-0 (IMT-2020 Requirements). The CPU frequency was randomly generated between 1000 MHz and 5000 MHz, as referenced in [90], which surveyed the CPU capacity of smartphones. To represent the different capacities of various types of devices, the CPU capacity was set between 0.5 and 2. Although there are detailed models for energy consumption from transmission and com-

putation, we simplify by using the total allowed training time due to the lack of comprehensive surveys on energy consumption. We list the setting in Table 3.

For data heterogeneity, five different non-IID distributions were considered:

1) **DATASIZESCORE**: There were 100 clients, each with correctly labeled and IID datasets but with different dataset sizes. The dataset size for each client increased by 100 samples. Under both MNIST and CIFAR-10, there were 10 labels, and each client had an equal number of each label.

2) **QUALITYSCORE**: There were 100 clients, each with a dataset size of 500 samples, correctly labeled but exhibiting different levels of label-skew. The number of labels each client had increased by one for every 10 clients: the first 10 clients had only 1 label, and the last 10 clients had 10 labels.

3) **MISLABELSCORE**: There were 100 clients, each with a dataset size of 500 samples and 10 labels. The mislabel rate increased by 10% per client, with the first 10 clients having a 0% mislabel rate and the last 10 clients having a 90% mislabel rate.

4) **DATASIZESCORE + MISLABELSCORE_TEN**: In addition to the quantity-skew as in DATASIZESCORE, , there were varying levels of label-skew in their datasets. the mislabel rate for client i was $(i/10) \times 10$ %.

5) **QUALITYSCORE + MISLABELSCORE_TEN**: In addition to the quality-skew as in QUALITYSCORE, there are varying levels of mislabeling in their datasets. the mislabel rate for the client with ID $(i/10) \times 10$ %.

Table 2: Learning setting

Notation	MNIST	CIFAR-10
Sample size	$28 \times 28 \times 1$ Bytes	$64 \times 64 \times 3$ Bytes
Dataset size D_i	500 samples (default)	500 samples (default)
selection number N_T	10	10
Task structure	Same as the model in FedAvg	Same as the model in FedAvg
Model size $\|\omega_i^t\|$	2.3 MB	3.4 MB
Learning rate γ_{loc}	0.005	0.005
Batch size B_{loc}	64	64
local iteration I_{loc}	10	10

Table 3: Hardware setting

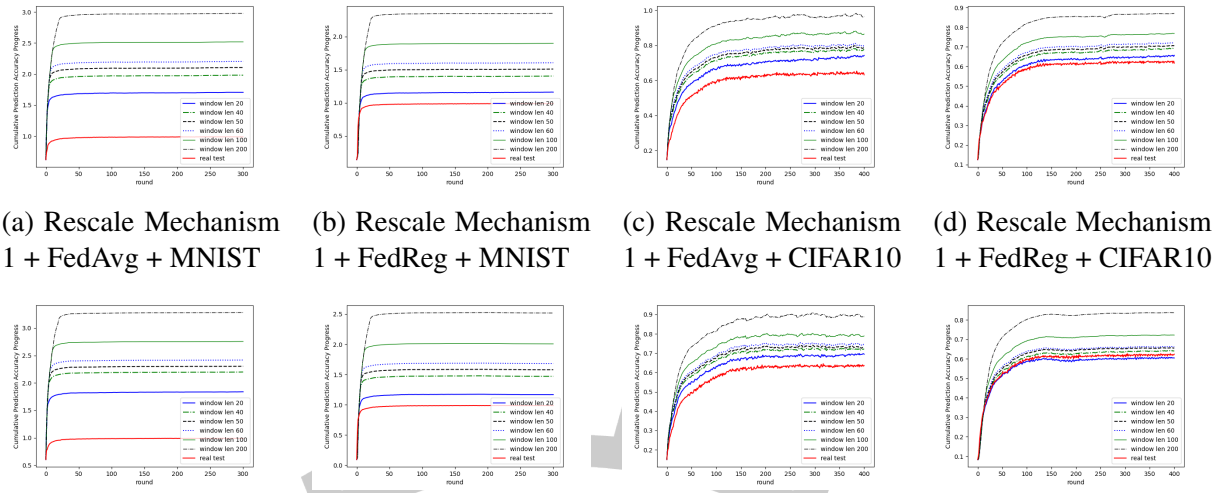
Notation	VALUE
transmission rate	[100 Mbps 10 Gbp]
CPU frequency f_i^t	[1000 MHz 5000 MHz]
CPU capacity C_i	[0.5 2]
remaining time λ_i^t	[10 100]

7.2 Rescale Mechanism & regression window length

In Section 6.2, we proposed two rescaling mechanisms to convert the evaluated Shapley value to the client's contribution in each round. To determine which mechanism provides a closer estimate of the true contribution, we conducted experiments using the two different rescaling mechanisms, under varying window lengths of the regression model. We then compare the accuracy progress trends of the true contributions with the predicted contributions.

In the experiment, three different non-IID feature datasets were considered **QUALITYSCORE**, **MISLABELSCORE**, and **DATASIZESCORE** in the MNIST and CIFAR-10 datasets. For each case distribution, we conducted experiments using both the FedAvg and FedReg algorithms to evaluate the effectiveness of the rescaling mechanisms under different client selection algorithms.

From the experimental results shown in Figs. 17, 18 and 19, we can observe that, in the case of the quantity-skew feature (Fig. 17), both rescale mechanisms exhibited similar trends in mea-



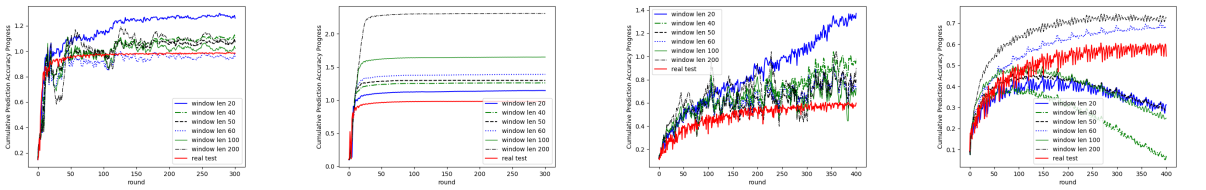
(a) Rescale Mechanism 1 + FedAvg + MNIST (b) Rescale Mechanism 1 + FedReg + MNIST (c) Rescale Mechanism 1 + FedAvg + CIFAR10 (d) Rescale Mechanism 1 + FedReg + CIFAR10

(e) Rescale Mechanism 2 + FedAvg + MNIST (f) Rescale Mechanism 2 + FedReg + MNIST (g) Rescale Mechanism 2 + FedAvg + CIFAR10 (h) Rescale Mechanism 2 + FedReg + CIFAR10

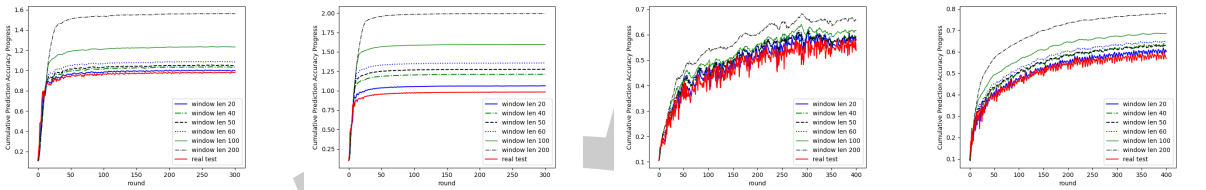
Figure 17: **Rescale Mechanism with Quantity-Skew**

sured accuracy and predicted accuracy progress. However, compared to rescale mechanism 1, rescale mechanism 2 showed a smaller error in measured accuracy upon reaching convergence. For the quality-skew and mislabel features (Figs. 18 and 19), there was a significant difference between the true measured accuracy and the predicted accuracy trends when using rescale mechanism 1. Notably, in Figs. 18d and 19d, the trends even moved in opposite directions. Additionally, under the random selection strategy (Figs. 18a, 18c, 19a, and 19c), the variance in predicted accuracy progress was large (with significant fluctuations). This large variance can cause the trade-off between accuracy and latency to be skewed, as the predicted accuracy scale is too large, leading to latency being overlooked. However, with rescale mechanism 2, the accuracy prediction and actual accuracy trends were more aligned. For example, in the CIFAR-10 results with the mislabel feature (Fig. 19g), as the accuracy decreased, the predicted accuracy also showed a downward trend. In contrast, with rescale mechanism 1 (Fig. 19c), the predicted accuracy continued to increase despite the actual accuracy decreased.

Furthermore, in the experiments, as the regression window length increased, the error between the predicted accuracy and actual accuracy grew. Conversely, shorter regression windows



(a) Rescale Mechanism 1 + FedAvg + MNIST (b) Rescale Mechanism 1 + FedReg + MNIST (c) Rescale Mechanism 1 + FedAvg + CIFAR10 (d) Rescale Mechanism 1 + FedReg + CIFAR10



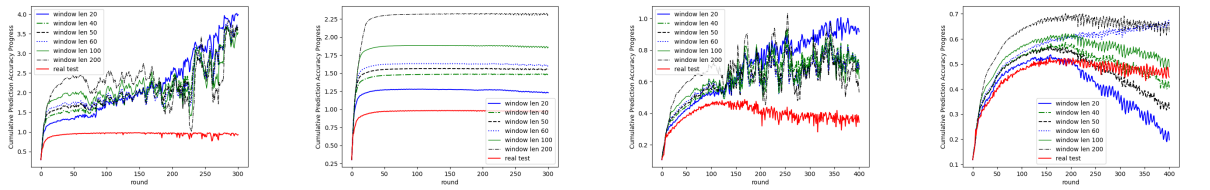
(e) Rescale Mechanism 2 + FedAvg + MNIST (f) Rescale Mechanism 2 + FedReg + MNIST (g) Rescale Mechanism 2 + FedAvg + CIFAR10 (h) Rescale Mechanism 2 + FedReg + CIFAR10

Figure 18: **Rescale Mechanism with label-Skew**

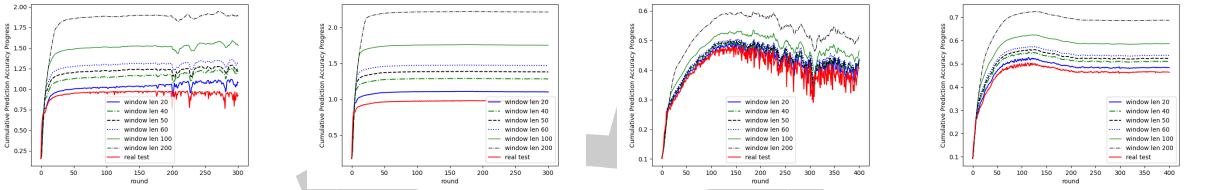
resulted in more accurate predictions. This supports our earlier hypothesis that different stages of the FL training process have varying performance and requirements. Therefore, our designed learnable prediction model should discard outdated historical information over time.

7.3 Fairness Coefficient

In Section 6.4, we introduce the fairness coefficients α_1 and α_2 to determine the size l_f of the fairness-aware potential set. Since this value is calculated based on the variance of all clients' quality scores, and the range of variance is from 0 to 0.25, we set α_1 to 4 to ensure that the value of Equation (2) falls within the range of 0 to 1. Next, to find the optimal α_2, we conduct experiments on the CIFAR-10 dataset under two distributions, QUALITYSCORE and MISLABELSCORE, testing different values of α_2 to identify the best one. From the experimental results, in the QUALITYSCORE distribution shown in Fig. 20a, α_2 values between 0.2 and 0.4 showed better utility. However, as α_2 increased beyond this range, the utility decreased. This is because fairness had a significant impact on the QUALITYSCORE distribution, and selecting only the clients with the best quality led to local optimization when α_2 was too large. In the

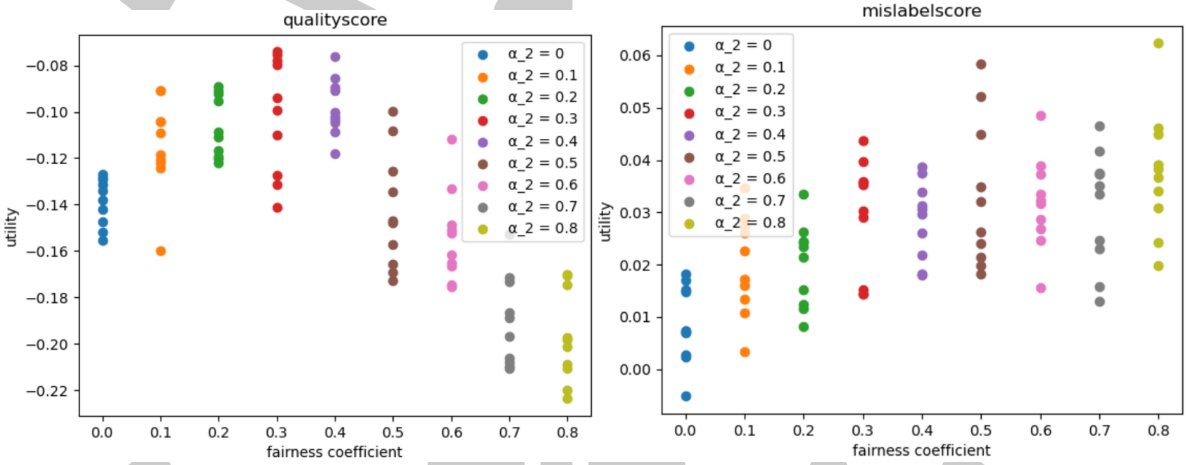


(a) Rescale Mechanism 1 + FedAvg + MNIST (b) Rescale Mechanism 1 + FedReg + MNIST (c) Rescale Mechanism 1 + FedAvg + CIFAR10 (d) Rescale Mechanism 1 + FedReg + CIFAR10



(e) Rescale Mechanism 2 + FedAvg + MNIST (f) Rescale Mechanism 2 + FedReg + MNIST (g) Rescale Mechanism 2 + FedAvg + CIFAR10 (h) Rescale Mechanism 2 + FedReg + CIFAR10

Figure 19: **Rescale Mechanism with mislabel**



(a) CIFAR-10 with QUALITYSCORE (b) CIFAR-10 with MISLABELSCORE

Figure 20: fairness coefficient

MISLABELSCORE distribution 20b, α_2 values between 0 and 0.2 showed poor utility, but starting from 0.3, the utility gradually increased with α_2. This was because mislabeled samples did not provide useful information, and emphasizing fairness could reduce accuracy performance. A larger fairness-aware potential set allows the server to select clients with lower latency, resulting in better utility. Although increasing α_2 in the MISLABELSCORE distribution improved utility, considering that such severe mislabeling situations are unlikely in real-world scenarios and to ensure the generalizability of our method, we use $\alpha_2 = 0.3$ in subsequent experiments.

7.4 Impact Under Different Non-IID Levels

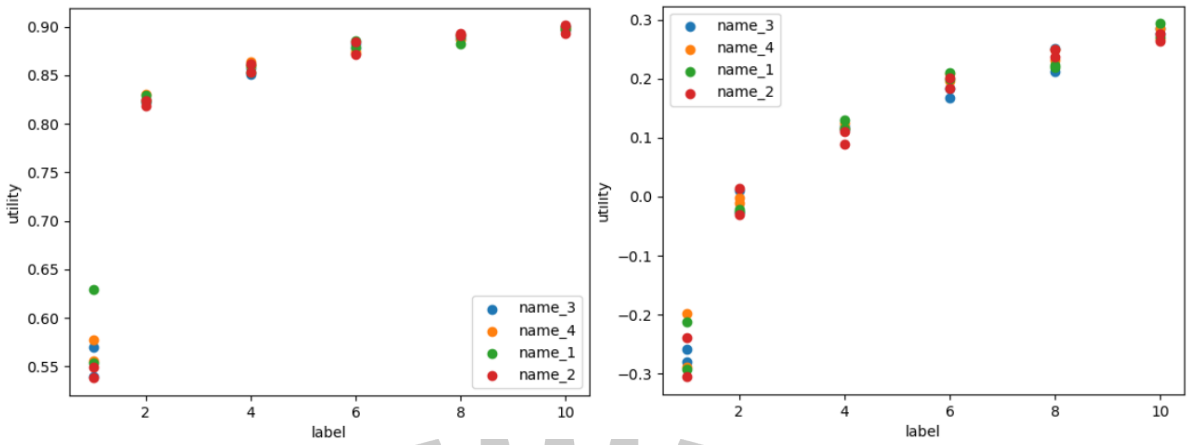
To evaluate how the non-IID level impacts the objective utility, we conducted experiments under two different non-IID features:

1) Label-Skew: There were 6 different levels of label-skew, with each client having 1, 2, 4, 6, 8, or 10 labels, respectively.

2) Mislabel Feature: There were 6 different levels of mislabeling, with each client having a mislabel rate of 0%, 10%, 20%, 30%, 40%, or 50%.

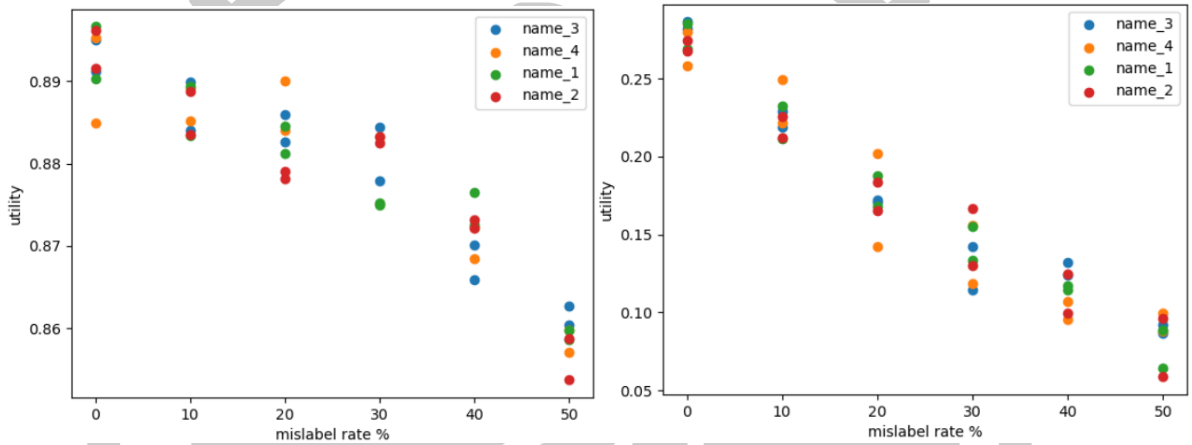
The result of the Label-Skew feature setting showed that the utility increased with the number of labels each client had, and there was a significant gap from 1 to 2. This result corresponds to the previous experiments shown in Figs. 4b and 4a. For the mislabel feature setting, where the utility decreased while the mislabel rate of each client increased. This result makes sense since the mislabel rate degrades the test accuracy and, thus, the utility that considers the accuracy.

The result of the Label-Skew feature setting showed that the utility increased with the number of labels each client had. This was because, with higher label-skew levels, the convergence rate decreased, resulting in lower achievable accuracy compared to lower label-skew levels. There was a significant gap in utility from 1 to 2 labels, which corresponds to the previous experiments shown in Figs. 4b and 4a. For the mislabel feature setting. The utility decreased as the mislabel rate of each client increased. This is logical because a higher mislabel rate degrades test accuracy, reducing the utility that considers accuracy. These results demonstrated that the non-IID level impacted the utility under our solution, decreasing utility as the mislabel level increased or label-skew decreased.



(a) MNIST (b) CIFAR-10

Figure 21: impact of the label-skew level



(a) MNIST (b) CIFAR-10

Figure 22: impact of the mislabel level

7.5 Comparisons With Other Approaches

To compare the efficiency of our algorithm in selecting clients to optimize the objective, we considered five different benchmark solutions. Below, we brief each benchmark algorithm. Note that in each algorithm, the number of selected clients N_T in each round is set to 10 for all rounds:

1) **FedAvg**: This is the first FL algorithm provided by Google. It selects clients randomly for each round without considering hardware and data distribution heterogeneity. The random selection aims to ensure an unbiased representation of clients, but it does not

account for differences in client capabilities or data characteristics.

2) **LATENCY_ONLY**: Based on the study [73], this approach focuses solely on hardware heterogeneity. The primary goal is to minimize training time by selecting the fastest clients for each round. This method prioritizes clients with the highest computational speed and most stable network connections, ensuring that slower clients do not drag down the training process.

3) **LYAPUNOV_OPT**: Based on the studies [82] and [79], this approach considers both accuracy and latency. It aims to ensure fairness in client selection to enhance model performance. This method uses the multi-armed bandit (MAB) algorithm to predict each client's latency and then solves the optimization problem of balancing latency and fairness using traditional optimization algorithms. However, in our study, we assume that the server knows the training latency of each client in advance for the next round, thus ignoring the MAB component.

4) **CumulativeSV**: Based on [91], which focuses on model performance. It evaluates each client's quality using the Cumulative Shapley value based on the GTG-Shapley value. This method selects clients based on their contributions to model performance, as determined by the Shapley value, aiming to maximize the overall effectiveness of the training process.

We compared them with five different distributions mentioned in Section 7.1 with the metric:

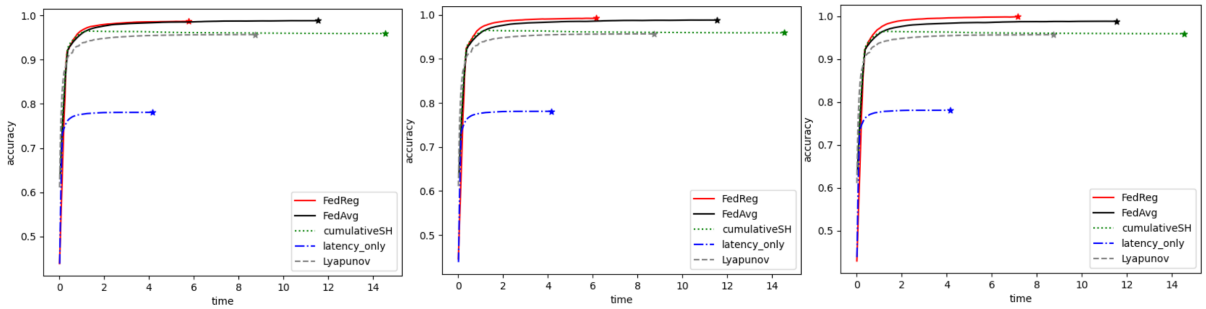
$$\text{utility} = \sum_{t \in T} \Delta \text{acc}_t - \delta L_t. \tag{7.1}$$

7.5.1 Evaluation in DATASIZESCORE

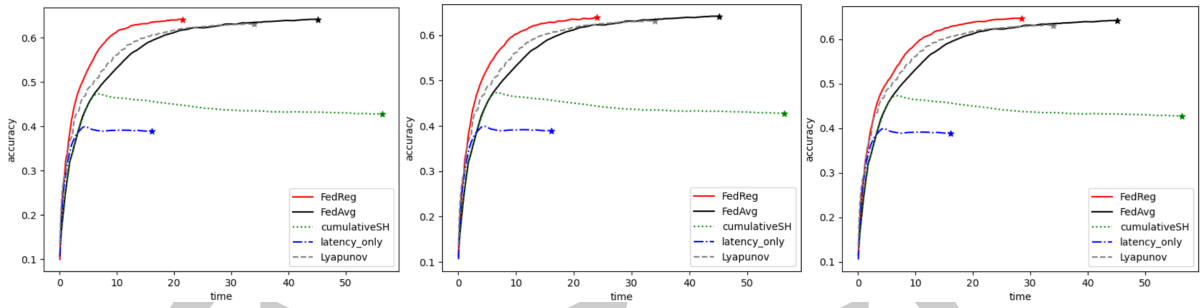
In the DATASIZESCORE distribution, which represents the quantity-skew non-IID feature, the results show that FedReg, and Lyapunov performed well in terms of latency because they explicitly consider latency in their algorithms. The latency_only algorithm achieved the shortest latency because it selects clients solely based on the shortest training latency, without considering accuracy improvements. However, Lyapunov performed worse than our solution because it incorporates a fairness term after the Lyapunov optimization, and the value of the coefficient for this term is unknown. Regarding test accuracy, latency_only performed the worst on both the CIFAR-10 and MNIST datasets. CumulativeSH showed acceptable performance on MNIST but not on CIFAR-10. The reason for this discrepancy is that CumulativeSH does not consider the fairness term, leading to degraded performance in more complex tasks like CIFAR-10. When considering utility, which takes into account both latency and accuracy, our solution achieved the best result among the five algorithms under three different considerations. Additionally, our results indicate that as the weight δ decreased, both accuracy and latency increased. This suggests that our solution effectively balanced the trade-off between accuracy and latency by adjusting the weight δ.

DATASIZESCORE, MNIST															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.929	0.987	5.769	0.873	0.988	11.547	0.814	0.959	14.556	0.740	0.781	4.142	0.870	0.957	8.743
0.005	0.961	0.992	6.164	0.930	0.988	11.547	0.886	0.959	14.556	0.760	0.781	4.142	0.913	0.957	8.743
0.001	0.991	0.999	7.154	0.977	0.988	11.547	0.945	0.959	14.556	0.777	0.781	4.142	0.948	0.957	8.743

DATASIZESCORE, CIFAR-10															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.427	0.642	21.515	0.190	0.642	45.155	-0.137	0.428	56.482	0.228	0.389	16.135	0.291	0.632	34.068
0.005	0.518	0.638	24.019	0.416	0.642	45.155	0.145	0.428	56.482	0.309	0.389	16.135	0.461	0.632	34.068
0.001	0.619	0.647	28.460	0.597	0.642	45.155	0.371	0.428	56.482	0.373	0.389	16.135	0.598	0.632	34.068



(a) $\delta = 0.01$, MNIST (b) $\delta = 0.005$, MNIST (c) $\delta = 0.001$, MNIST



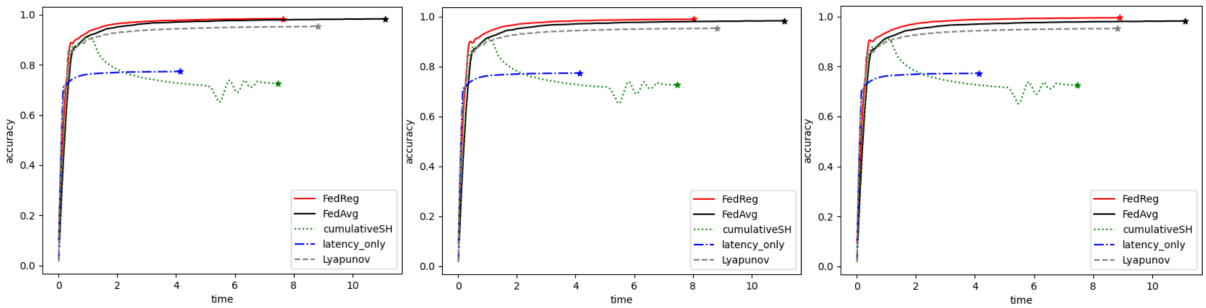
(d) $\delta = 0.01$, CIFAR-10 (e) $\delta = 0.005$, CIFAR-10 (f) $\delta = 0.001$, CIFAR-10

Figure 23: evaluation in **DATASIZESCORE**

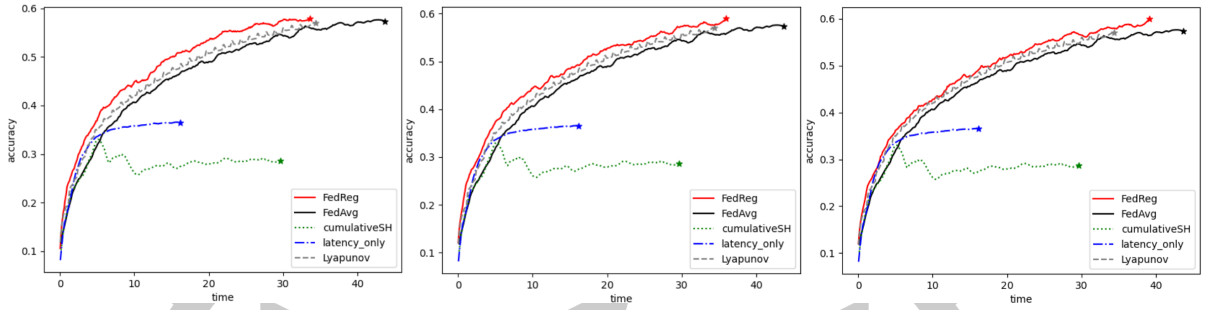
7.5.2 Evaluation in QUALITYSCORE

In the QUALITYSCORE distribution, which features label-skew non-IID data, fairness was the main factor affecting accuracy. The results indicate that the test accuracy was similar among the three algorithms: FedAvg, Lyapunov, and our FreReg. In contrast, CumulativeSH and latency_only showed significantly worse accuracy due to their lack of fairness-awareness. Our algorithm outperformed the others regarding utility, which considers both latency and accuracy. The trend is consistent with the results observed in the DATASIZESCORE distribution, where accuracy and latency increased with more emphasis on accuracy.

QUALITYSCORE, MNIST															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.907	0.983	7.647	0.871	0.982	11.122	0.651	0.726	7.452	0.732	0.773	4.142	0.864	0.952	8.813
0.005	0.948	0.989	8.027	0.927	0.982	11.122	0.688	0.726	7.452	0.752	0.773	4.142	0.908	0.952	8.813
0.001	0.986	0.995	8.899	0.971	0.982	11.122	0.718	0.726	7.452	0.769	0.773	4.142	0.944	0.952	8.813



(a) $\delta = 0.01$, MNIST (b) $\delta = 0.005$, MNIST (c) $\delta = 0.001$, MNIST



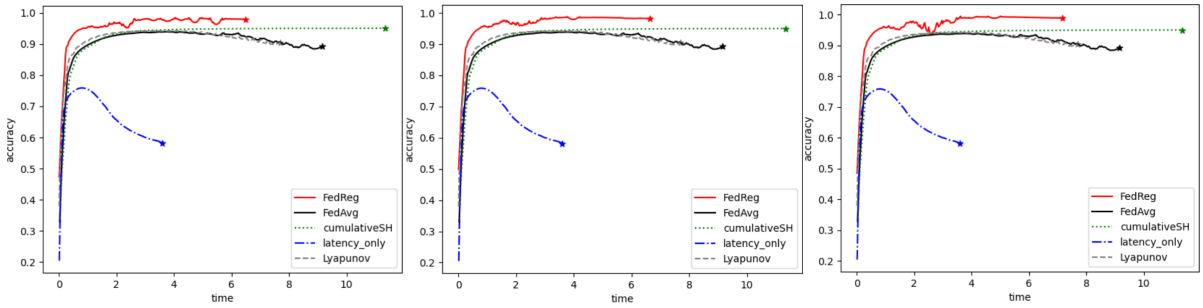
(d) $\delta = 0.01$, CIFAR-10 (e) $\delta = 0.005$, CIFAR-10 (f) $\delta = 0.001$, CIFAR-10

Figure 24: evaluation in **QUALITYSCORE**

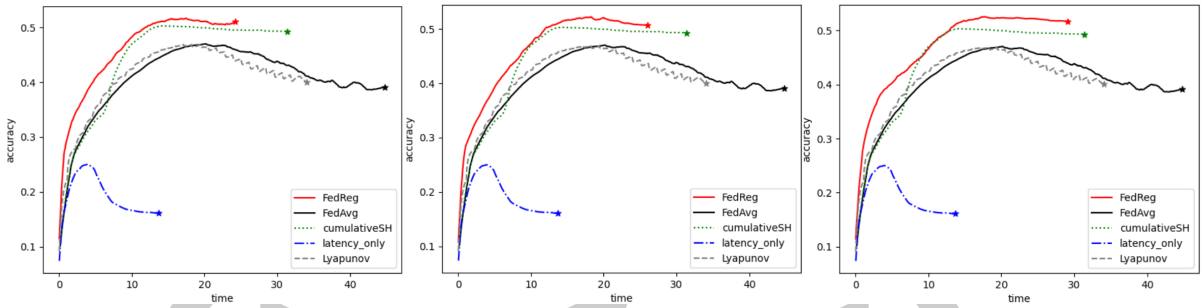
QUALITYSCORE, CIFAR-10															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.239	0.575	33.607	0.138	0.575	43.737	-0.012	0.285	29.666	0.205	0.366	16.135	0.221	0.565	34.394
0.005	0.402	0.582	35.951	0.357	0.575	43.737	0.136	0.285	29.666	0.285	0.366	16.135	0.393	0.565	34.394
0.001	0.551	0.590	39.124	0.532	0.575	43.737	0.255	0.285	29.666	0.350	0.366	16.135	0.530	0.565	34.394

7.5.3 Evaluation in MISLABELSCORE

In the MISLABELSCORE distribution, which features mislabeled non-IID data, CumulativeSH and our FedReg achieved better accuracy performance. As mentioned in Section 5.1.3, including mislabeled samples provides no benefit. In this scenario, selecting clients with many mislabeled samples degraded performance. The algorithms FedAvg and Lyapunov, which incorporate fairness, tended to select these clients during training, reducing overall accuracy. Therefore, our FedReg and CumulativeSH, which effectively avoided clients with many mislabeled samples, showed superior performance in this distribution, and there was no significant accuracy drop during the training like in the other three algorithms. In terms of utility, our algorithm



(a) $\delta = 0.01$, MNIST (b) $\delta = 0.005$, MNIST (c) $\delta = 0.001$, MNIST



(d) $\delta = 0.01$, CIFAR-10 (e) $\delta = 0.005$, CIFAR-10 (f) $\delta = 0.001$, CIFAR-10

Figure 25: evaluation in **MISLABELSCORE**

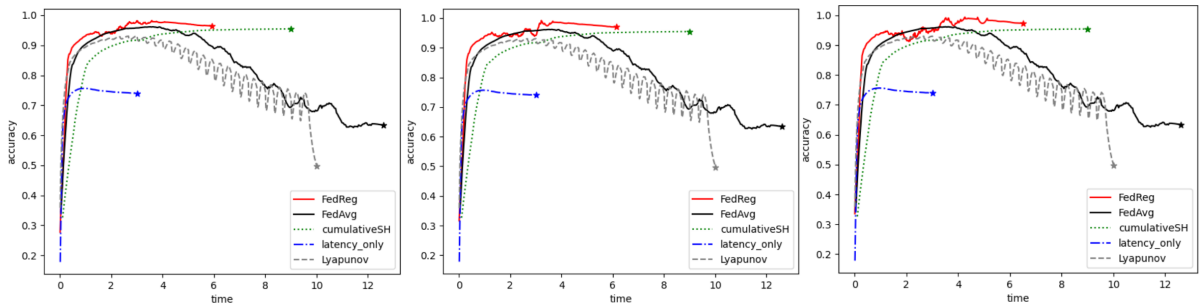
performed far better than the others because it considers both latency and mislabeling simultaneously. The trend is consistent with the results observed in the DATASIZESCORE distribution, where accuracy and latency increased with more emphasis on accuracy.

MISLABELSCORE, MNIST															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.914	0.979	6.477	0.796	0.888	9.155	0.837	0.950	11.353	0.548	0.584	3.596	0.824	0.901	7.704
0.005	0.949	0.982	6.632	0.842	0.888	9.155	0.893	0.950	11.353	0.566	0.584	3.596	0.862	0.901	7.704
0.001	0.982	0.989	7.171	0.878	0.888	9.155	0.939	0.950	11.353	0.581	0.584	3.596	0.893	0.901	7.704

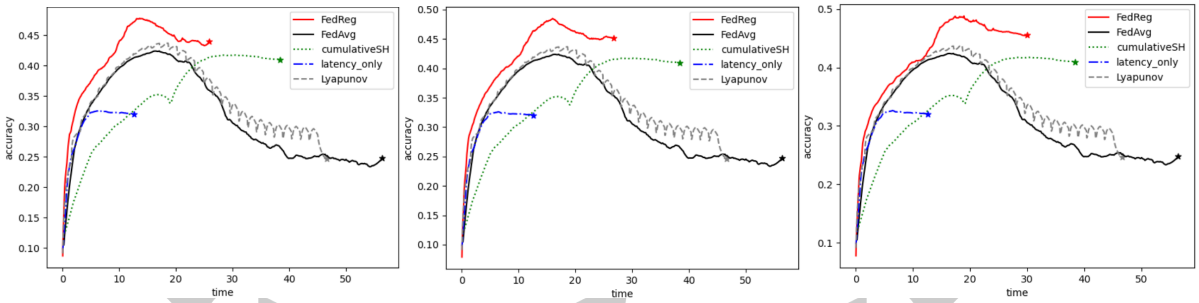
MISLABELSCORE, CIFAR-10															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.265	0.507	24.231	-0.059	0.389	44.845	0.179	0.493	31.400	0.024	0.161	13.716	0.063	0.404	34.069
0.005	0.377	0.507	25.983	0.165	0.389	44.845	0.336	0.493	31.400	0.093	0.161	13.716	0.233	0.404	34.069
0.001	0.488	0.517	29.099	0.344	0.389	44.845	0.462	0.493	31.400	0.147	0.161	13.716	0.369	0.404	34.069

7.5.4 Evaluation in DATASIZESCORE+MISLABELSCORE_TEN

In the DATASIZESCORE+MISLABELSCORE_TEN distribution, which features quantity-skew and mislabeled non-IID data, both FedReg and CumulativeSH achieved better perfor-



(a) $\delta = 0.01$, MNIST (b) $\delta = 0.005$, MNIST (c) $\delta = 0.001$, MNIST



(d) $\delta = 0.01$, CIFAR-10 (e) $\delta = 0.005$, CIFAR-10 (f) $\delta = 0.001$, CIFAR-10

Figure 26: evaluation in **DATASIZESCORE+MISLABELSCORE_TEN**

mance in terms of utility and accuracy. However, in the MNIST dataset with $\delta = 0.01$, the latency_only algorithm achieved the highest utility. The trend is consistent with the results observed in the DATASIZESCORE distribution, where accuracy and latency increased with more emphasis on accuracy.

DATASIZESCORE+MISLABELSCORE_TEN, MNIST															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.906	0.965	5.926	0.510	0.636	12.619	0.864	0.954	9.003	0.710	0.740	3.003	0.587	0.687	10.012
0.005	0.939	0.970	6.136	0.573	0.636	12.619	0.909	0.954	9.003	0.725	0.740	3.003	0.637	0.687	10.012
0.001	0.968	0.974	6.499	0.623	0.636	12.619	0.945	0.954	9.003	0.737	0.740	3.003	0.677	0.687	10.012

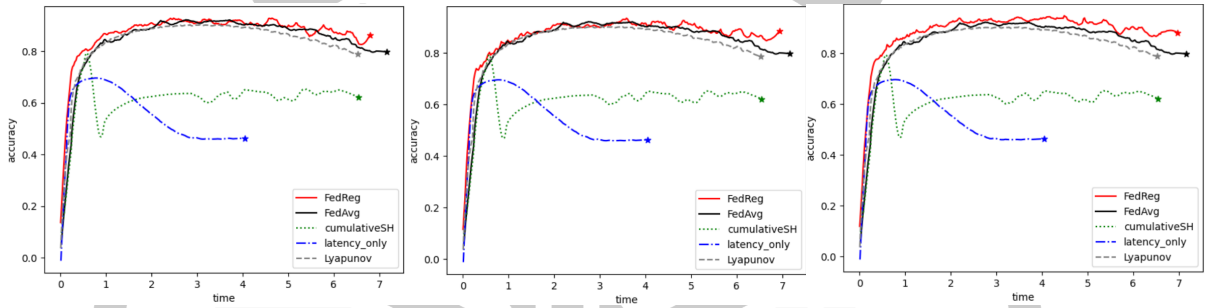
DATASIZESCORE+MISLABELSCORE_TEN, CIFAR-10															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.181	0.438	25.676	-0.326	0.238	56.416	0.019	0.407	38.806	0.192	0.318	12.616	-0.180	0.285	46.546
0.005	0.322	0.454	26.372	-0.044	0.238	56.416	0.213	0.407	38.686	0.255	0.318	12.616	0.053	0.285	46.546
0.001	0.427	0.457	30.423	0.182	0.238	56.416	0.368	0.407	38.653	0.306	0.318	12.616	0.239	0.285	46.546

7.5.5 Evaluation in QUALITYSCORE+MISLABELSCORE_TEN

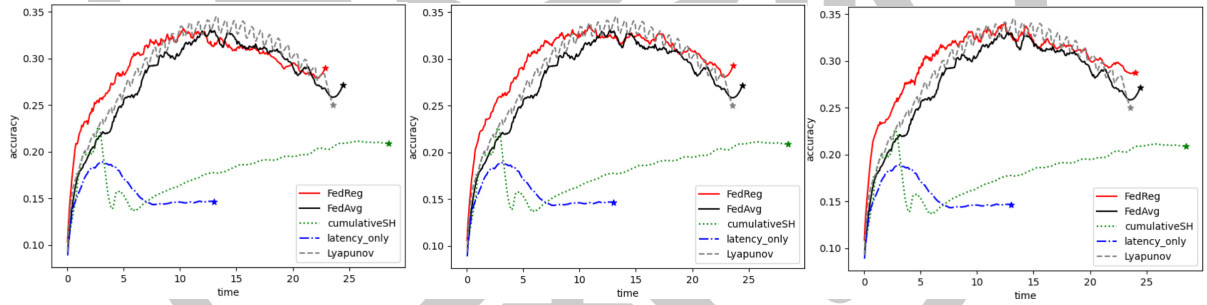
The last distribution was QUALITYSCORE+MISLABELSCORE_TEN, which features the label-skew and mislabeled non-iid data. The algorithms FedAvg, FedReg, and Lyapunov performed similarly in accuracy and training latency. However, our algorithm, FedReg, got a little better than the other two in accuracy. Concerning the utility, our algorithm was the best in all settings. The trend is consistent with the results observed in the DATASIZESCORE distribution, where accuracy and latency increased with more emphasis on accuracy.

QUALITYSCORE+MISLABELSCORE_TEN, MNIST															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.788	0.856	6.778	0.729	0.801	7.167	0.520	0.585	6.526	0.418	0.458	4.041	0.736	0.801	6.523
0.005	0.845	0.879	6.891	0.765	0.801	7.167	0.552	0.585	6.526	0.438	0.458	4.041	0.768	0.801	6.523
0.001	0.886	0.892	6.966	0.794	0.801	7.167	0.578	0.585	6.526	0.454	0.458	4.041	0.794	0.801	6.523

QUALITYSCORE+MISLABELSCORE_TEN, CIFAR-10															
	FedReg			FedAvg			CumulativeSH			latency_only			Lyapunov		
δ	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency	score	acc	latency
0.01	0.057	0.287	22.985	0.020	0.264	24.458	-0.083	0.202	28.522	0.015	0.145	13.013	0.040	0.276	23.592
0.005	0.174	0.291	23.441	0.142	0.264	24.458	0.059	0.202	28.522	0.080	0.145	13.013	0.158	0.276	23.592
0.001	0.267	0.291	23.936	0.240	0.264	24.458	0.173	0.202	28.522	0.132	0.145	13.013	0.252	0.276	23.592



(a) $\delta = 0.01$, MNIST (b) $\delta = 0.005$, MNIST (c) $\delta = 0.001$, MNIST



(d) $\delta = 0.01$, CIFAR-10 (e) $\delta = 0.005$, CIFAR-10 (f) $\delta = 0.001$, CIFAR-10

Figure 27: evaluation in **QUALITYSCORE+MISLABELSCORE_TEN**

Chapter 8. Conclusions

In this work, we have addressed the client selection problem in Federated Learning (FL) with data and hardware heterogeneity by focusing on key factors such as model accuracy and training latency. Our proposed solution accounts for a broader range of non-IID features, distinguishing it from previous studies that typically consider only a subset of these features. To maximize utility in terms of accuracy and training latency, we first apply a privacy-aware measurement method to quantify the non-IID level of each client. This information is then used as input for our accuracy contribution predictor, which forecasts each client's contribution to model accuracy. To capture varying contribution performances at different FL stages, we employ a Shapley value-based evaluation conducted at the end of each global training round as the learning target and use a ridge regression model to predict accuracy contributions. We then determine the optimal set of clients for each round by solving problem P2 from the fairness-aware potential set, substituting actual accuracy contributions with the predictions generated by our model. We conducted several experiments to validate our approach. The experiment results demonstrated that a learnable predictive model prioritizing recent interactions outperforms models considering all historical data, and there is a small gap between actual and predicted accuracy progress trends under rescale mechanism 2. Additional experiments also showed that utility increased as the label-skew and mislabel levels decreased, owing to the availability of higher-quality potential clients. Finally, we compared our proposed approach with four bench-

marks FedAvg, LATENCY_ONLY, LYAPUNOV_OPT and CumulativeSV under five different non-IID features. Our approach outperformed these methods in most cases because it considers more non-IID features and models training latency, demonstrating its effectiveness.

Concerning future work, several directions could be explored. While we have proposed an effective quality score evaluation method to quantify a client's mislabeling and label-skew levels simultaneously, the current fairness-aware algorithm determines the importance of fairness based on the variance between the client's quality score. This may lead to performance degradation because the difficulty distinguishing the score level is more influenced by mislabeling or label-skew. Therefore, developing methods to distinguish and quantify these non-IID features separately could enhance the performance of our approach. Secondly, in real-world applications, data owners incur costs, such as the cost of data collection and energy consumption. This highlights the need for an incentive mechanism design. Our proposed method offers a framework for modeling the relationship between various non-IID features and contribution progress. Because of the small-gap accuracy, prediction could be further extended to incorporate incentive mechanisms.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] "Regulation (EU) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," Official Journal of the European Union, L 119, 1-88, 2016.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of Machine Learning Research*, vol. 54, pp. 1273–1282, 2017.

[4] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, August 2020.

[5] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *Proceedings of Machine Learning Research*, vol. 97, pp. 4615–4625, 2019.

[6] K. Wei, J. Li, C. Ma, M. Ding, S. Wei, F. Wu, G. Chen, and T. Ranbaduge, "Vertical federated learning: Challenges, methodologies and experiments," *arXiv:2202.04309*, 2022.

[7] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3615–3634, July 2024.

[8] Y. Shi, Y. Zhang, and K. B. Letaief, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2036–2051, November 2020.

[9] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, November 2020.

[10] Y. Shen, Y. Qu, C. Dong, F. Zhou, and Q. Wu, "Joint training and resource allocation optimization for federated learning in UAV swarm," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2272–2284, February 2023.

[11] T. Li, "On the convergence of local descent methods in federated learning," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 5097–5110, December 2019.

[12] K. Hsieh, "Local adaptivity in federated learning: Convergence and consistency," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3577–3590, June 2021.

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

[14] L. Li, Y. Fan, and K.-Y. Lin, "A survey on federated learning," in *IEEE 16th International Conference on Control & Automation*, Singapore, October 2020.

[15] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv:2003.02133*, Mar 2020.

[16] P. M. Mammen, "Federated learning: Opportunities and challenges," *arXiv:2101.05428*, January 2021.

[17] S. Abdulrahman, H. Tout, H. Ould-Slimane, and A. Mourad, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, April 2021.

[18] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *IEEE 38th International Conference on Data Engineering (ICDE)*, Kuala Lumpur, Malaysia, May 2022.

[19] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-IID data in federated learning," *Future Generation Computer Systems*, vol. 135, pp. 244–258, October 2022.

[20] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *arXiv:1806.00582*, July 2022.

[21] L. Qu, N. Balachandar, and D. L. Rubin, "An experimental study of data heterogeneity in federated learning methods for medical imaging," *arXiv:2107.08371*, July 2021.

[22] Y. Huang and C. Hu, "Toward data heterogeneity of federated learning," *arXiv:2212.08944*, December 2022.

[23] A. Alwosheel, S. van Cranenburgh, and C. G. Chorus, "Is your dataset big enough? sample size requirements when using artificial neural networks for discrete choice analysis," *Journal of Choice Modelling*, vol. 28, pp. 167–182, September 2018.

[24] D. Brain and G. I. Webb, "On the effect of data set size on bias and variance in classification learning," in *Proceedings of the 4th Australian Conference on Artificial Intelligence*. Berlin, Heidelberg: Springer, 1999, pp. 117–120.

[25] A. Althnian, D. AlSaeed, H. Al-Baity, A. Samha, A. Bin Dris, N. Alzakari, A. Abou Elwafa, and H. Kurdi, "Impact of dataset size on classification performance: An empirical evaluation in the medical domain," *Applied Sciences*, vol. 11, no. 2, p. 796, 2021.

[26] A. Bailly, C. Blanc, É. Francis, T. Guillotin, F. Jamal, B. Wakim, and P. Roy, "Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models," *Computer Methods and Programs in Biomedicine*, vol. 213, p. 106504, January 2022.

[27] J. Prusa, T. M. Khoshgoftaar, and N. Seliya, "The effect of dataset size on training tweet sentiment classifiers," in *IEEE 14th International Conference on Machine Learning and Applications*, Miami, FL, USA, December 2015, pp. 1–6.

[28] Z. Yang, M. Chen, W. Saad, and C. S. Hong, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, March 2021.

[29] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, February 2021.

[30] S. A. Tijani, X. Ma, R. Zhang, F. Jiang, and R. Doss, "Federated learning with extreme label skew: A data extension approach," in *IEEE International Joint Conference on Neural Networks*, Shenzhen, China, July 2021.

[31] Y. Diao, Q. Li, and B. He, "Towards addressing label skews in one-shot federated learning," in *Proceedings of the International Conference on Learning Representations*, 2023.

[32] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 630–641, March 2022.

[33] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv:1912.00818*, December 2019.

[34] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, J. Jiang, and C. Zhang, "Multi-center federated learning: Clients clustering for better personalization," *World Wide Web*, vol. 26, pp. 481–500, February 2023.

[35] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 35–44, May 2020.

[36] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-IID data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, pp. 7865–7873, July 2020.

[37] B. Gong, T. Xing, Z. Liu, W. Xi, and X. Chen, "Adaptive client clustering for efficient federated learning over non-iid and imbalanced data," *IEEE Transactions on Big Data*, pp. 1–1, April 2022, early Access.

[38] M. S. Islam, S. Javaherian, F. Xu, X. Yuan, L. Chen, and N.-F. Tzeng, "FedClust: Optimizing federated learning on non-IID data through weight-driven client clustering," in *IEEE*

International Parallel and Distributed Processing Symposium Workshops, San Francisco, CA, USA, May 2024.

[39] H.-Y. Hsu, K. H. Keoy, J.-R. Chen, H.-C. Chao, and C.-F. Lai, "Personalized federated learning algorithm with adaptive clustering for non-IID IoT data incorporating multi-task learning and neural network model characteristics," *Sensors*, vol. 23, no. 22, p. 9016, November 2023.

[40] Z. Cheng, M. Min, M. Liwang, Z. Gao, and Lianf, "Joint client selection and task assignment for multi-task federated learning in MEC networks," in *IEEE Global Communications Conference*, Madrid, Spain, December 2021.

[41] J. Xu, H. Wang, and L. Chen, "Bandwidth allocation for multiple federated learning services in wireless edge networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 4, pp. 2534–2546, April 2022.

[42] M. N. H. Nguyen, N. H. Tran, Y. K. Tun, and Z. H, "Toward multiple federated learning services resource sharing in mobile edge networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 541–555, January 2023.

[43] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proceedings of the 10th International Symposium on Information and Communication Technology*, Hanoi, Viet Nam, December 2019, pp. 273–279.

[44] N. Bhuyan and S. Moharir, "Multi-model federated learning," in *14th International Conference on COMmunication Systems NETworks*, Bangalore, India, January 2022.

[45] N. H. Nguyen, P. L. Nguyen, T. D. Nguyen, T. T. Nguyen, D. L. Nguyen, T. H. Nguyen, H. H. Pham, and T. N. Truong, "FedDRL: Deep reinforcement learning-based adaptive ag-

gregation for non-IID data in federated learning," in *Proceedings of the 51st International Conference on Parallel Processing*, Bordeaux, France, 2023, pp. 1–11.

[46] T. Zhang, K.-Y. Lam, J. Zhao, and J. Feng, "Joint device scheduling and bandwidth allocation for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, pp. 1–1, July 2023, early Access.

[47] T. Zhang, K.-Y. Lam, and J. Zhao, "Deep reinforcement learning based scheduling strategy for federated learning in sensor-cloud systems," *Future Generation Computer Systems*, vol. 144, pp. 219–229, July 2023.

[48] W. Huang, T. Li, D. Wang, S. Du, and J. Zhang, "Fairness and accuracy in federated learning," *arXiv:2012.10069*, December 2020.

[49] J. Zhang, S. Guo, Z. Qu, D. Zeng, Y. Zhan, Q. Liu, and R. Akerkar, "Adaptive federated learning on non-IID data with resource constraint," *IEEE Transactions on Computers*, vol. 71, no. 7, pp. 1655–1667, July 2022.

[50] W. Zhang, D. Yang, W. Wu, H. Peng, N. Zhang, H. Zhang, and X. Shen, "Optimizing federated learning in distributed industrial IoT: A multi-agent approach," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3688–3703, December 2021.

[51] G. Rjoub, O. A. Wahab, J. Bentahar, R. Cohen, and A. S. Bataineh, "Trust-augmented deep reinforcement learning for federated learning client selection," *Information Systems Frontiers*, pp. 1–18, July 2022.

[52] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *IEEE International Parallel and Distributed Processing Symposium*, New Orleans, LA, USA, May 2020.

[53] Z. Guan, Z. Wang, and X. Wang, "Deep reinforcement learning based efficient access scheduling algorithm with an adaptive number of devices for federated learning IoT systems," *Internet of Things*, vol. TBD, p. 100980, October 2023.

[54] Y. Deng, F. Lyu, J. Ren, and H. Wu, "Auction: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1996–2009, August 2022.

[55] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *IEEE INFOCOM*, Toronto, ON, Canada, July 2020, pp. 1–10.

[56] T. Zhao, F. Li, and L. He, "DRL-based joint resource allocation and device orchestration for hierarchical federated learning in NOMA-enabled industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 6, pp. 7468–7479, June 2023.

[57] J. Zheng, K. Li, N. Mhaisen, W. Ni, E. Tovar, and M. Guizani, "Exploring deep reinforcement learning-assisted federated learning for online resource allocation in privacy-persevering edge IoT," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 099–21 110, November 2022.

[58] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv:1808.04866*, July 2020.

[59] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with KPCA and k-means," in *Proceedings of the 2021 8th International Conference on Dependable Systems and Their Applications*, Yinchuan, China, August 2021, pp. 1–6.

[60] S. Duan, C. Liu, Z. Cao, X. Jin, and P. Han, "Fed-DR-Filter: Using global data representation to reduce the impact of noisy labels on the performance of federated learning," *Future Generation Computer Systems*, vol. 137, pp. 336–348, December 2022.

[61] S. Yang, H. Park, J. Byun, and C. Kim, "Robust federated learning with noisy labels," *IEEE Intelligent Systems*, vol. 37, no. 2, pp. 35–43, March-April 2022.

[62] J. Li, G. Li, H. Cheng, Z. Liao, and Y. Yu, "FedDiv: Collaborative noise filtering for federated learning with noisy labels," *arXiv:2312.12263*, February 2024.

[63] J. Xu, Z. Chen, T. Q. Quek, and K. F. E. Chong, "FedCorr: Multi-stage federated learning for label noise correction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA, June 2022.

[64] B. Zeng, X. Yang, Y. Chen, H. Yu, and Y. Zhang, "CLC: A consensus-based label correction approach in federated learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 5, pp. 1–23, June 2022.

[65] J. Zhang, D. Lv, Q. Dai, F. Xin, and F. Dong, "Noise-aware local model training mechanism for federated learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 4, pp. 1–22, June 2023.

[66] X. Fang and M. Ye, "Robust federated learning with noisy and heterogeneous clients," in *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA, June 2022.

[67] Y. Chen, X. Yang, X. Qin, H. Yu, and Z. Shen, "Focus: Dealing with label quality disparity in federated learning," in *Federated Learning: Privacy and Incentive*. Springer, 2020, pp. 153–167.

[68] M. Yang, H. Qian, X. Wang, Y. Zhou, and H. Zhu, "Client selection for federated learning with label noise," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2193–2197, February 2022.

[69] V. Tsouvalas, A. Saeed, T. Ozcelebi, and N. Meratnia, "Labeling chaos to learning harmony: Federated learning with noisy labels," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 2, pp. 1–26, February 2024.

[70] K. Pfeiffer, M. Rapp, R. Khalili, and J. Henkel, "Federated learning for computationally constrained heterogeneous devices: A survey," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–27, July 2023.

[71] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *Proceedings of the Web Conference*, Ljubljana, Slovenia, April 2021, pp. 935–946.

[72] D. Chen, C. S. Hong, L. Wang, and Y. Zh, "Matching-theory-based low-latency scheme for multitask federated learning in MEC networks," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 415–11 426, July 2021.

[73] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE International Conference on Communications*, Shanghai, China, May 2019.

[74] W. Yang, W. Xiang, Y. Yang, and P. Cheng, "Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1884–1893, February 2023.

[75] C. Battiloro, P. D. Lorenzo, M. Merluzzi, and S. Barbarossa, "Lyapunov-based optimization of edge resources for energy-efficient adaptive federated learning," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 265–280, March 2023.

[76] Y. Shi, H. Yu, and C. Leung, "Towards fairness-aware federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–17, April 2023, early Access.

[77] T. Huang, W. Lin, L. Shen, K. Li, and A. Y. Zomaya, "Stochastic client selection for federated learning with volatile clients," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20 055–20 070, October 2022.

[78] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, "A fairness-aware incentive scheme for federated learning," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, New York, NY, USA, February 2020, pp. 393–399.

[79] T. Huang, W. Lin, W. Wu, L. He, and K. Li, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552–1564, July 2021.

[80] H. Zhu, Y. Zhou, H. Qian, Y. Shi, X. Chen, and Y. Yang, "Online client selection for asynchronous federated learning with fairness consideration," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2493–2506, April 2023.

[81] D. Shi, L. Li, T. Ohtsuki, M. Pan, Z. Han, and H. V. Poor, "Make smart decisions faster: Deciding D2D resource allocation via Stackelberg game guided multi-agent deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4426–4438, June 2022.

[82] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, and H. H. Yang, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7108–7123, July 2020.

[83] D. B. Ami, K. Cohen, and Q. Zhao, "Client selection for generalization in accelerated federated learning: A bandit approach," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Rhodes Island, Greece, May 2023.

[84] X. Chang, S. M. Ahmed, S. V. Krishnamurthy, B. Guler, A. Swami, S. Oymak, and A. K. Roy-Chowdhury, "FLASH: Federated learning across simultaneous heterogeneities," *arXiv:2402.08769*, February 2024.

[85] B. Coll-Perales, M. C. Lucas-Estañ, T. Shimizu, and J. Gozalvez, "End-to-end V2X latency modeling and analysis in 5G networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 5094–5109, November 2023.

[86] M. C. Lucas-Estañ, B. Coll-Perales, T. Shimizu, J. Gozalvez, T. Higuchi, S. Avedisov, O. Altintas, and M. Sepulcre, "An analytical latency model and evaluation of the capacity of 5G NR to support V2X services using V2N2V communications," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 2293–2306, February 2023.

[87] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs. Available: http://yann.lecun.com/exdb/mnist*, 2010.

[88] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[89] J. Huang, R. Talbi, Z. Zhao, S. Boucchenak, L. Y. Chen, and S. Roos, "An exploratory analysis on users' contributions in federated learning," in *IEEE International Conference*

on Trust, Privacy and Security in Intelligent Systems and Applications, Atlanta, GA, USA, October 2020.

[90] P. Surana, N. Madhani, and T. Gopalakrishnan, "A comparative study on the recent smart mobile phone processors," in *7th International Conference on Smart Structures and Systems*, Chennai, India, July 2020, pp. 1–6.

[91] P. Singhal, S. R. Pandey, and P. Popovski, "Greedy Shapley client selection for communication-efficient federated learning," *IEEE Networking Letters*, vol. 6, no. 2, pp. 134–138, June 2024.