sensors

MDPI

Article

Adaptive Dataset Management Scheme for Lightweight Federated Learning in Mobile Edge Computing

Jingyeom Kim †,‡, Juneseok Bang † and Joohyung Lee *

School of Computing, Gachon University, Seongnam 13120, Republic of Korea; kimo1113@gachon.ac.kr (J.K.); dipsy1234@gachon.ac.kr (J.B.)

* Correspondence: j17.lee@gachon.ac.kr

† These authors contributed equally to this work as co-first authors.

‡ Current address: Advanced Research Team, NHN Cloud Corp, Seongnam 13487, Republic of Korea.

Abstract: Federated learning (FL) in mobile edge computing has emerged as a promising machine-learning paradigm in the Internet of Things, enabling distributed training without exposing private data. It allows multiple mobile devices (MDs) to collaboratively create a global model. FL not only addresses the issue of private data exposure but also alleviates the burden on a centralized server, which is common in conventional centralized learning. However, a critical issue in FL is the imposed computing for local training on multiple MDs, which often have limited computing capabilities. This limitation poses a challenge for MDs to actively contribute to the training process. To tackle this problem, this paper proposes an adaptive dataset management (ADM) scheme, aiming to reduce the burden of local training on MDs. Through an empirical study on the influence of dataset size on accuracy improvement over communication rounds, we confirm that the amount of dataset has a reduced impact on accuracy gain. Based on this finding, we introduce a discount factor that represents the reduced impact of the size of the dataset on the accuracy gain over communication rounds. To address the ADM problem, which involves determining how much the dataset should be reduced over classes while considering both the proposed discounting factor and Kullback–Leibler divergence (KLD), a theoretical framework is presented. The ADM problem is a non-convex optimization problem. To solve it, we propose a greedy-based heuristic algorithm that determines a suboptimal solution with low complexity. Simulation results demonstrate that our proposed scheme effectively alleviates the training burden on MDs while maintaining acceptable training accuracy.

Keywords: federated learning; mobile edge computing; dataset management

Citation: Kim, J.; Bang, J.; Lee, J. Adaptive Dataset Management Scheme for Lightweight Federated Learning in Mobile Edge Computing. *Sensors* **2024**, *24*, 2579. https://doi.org/10.3390/s24082579

Academic Editor: Tian Wang

Received: 3 March 2024
Revised: 3 April 2024
Accepted: 10 April 2024
Published: 18 April 2024

1. Introduction

With the success of deep learning (DL) and the rich storage and computing capabilities of mobile devices (MDs), many applications utilizing DL with the collected data from MDs, such as face recognition, recommendation systems, and human activity recognition, have become widely employed. Traditionally, the centralized learning method has been used for training DL models, which requires collecting all raw data from MDs by sending raw data to a remote cloud server. However, this approach raises concerns about MDs' privacy and potential data misuse [1]. To address these challenges, a promising distributed learning framework called federated learning (FL) has emerged [2,3]. FL allows for the construction of global DL models using only local model weights trained from MDs with their local datasets. This approach helps alleviate privacy concerns and distributes the training burden across multiple MDs at a centralized server. Nevertheless, efficient management of FL still faces several challenging issues due to its distributed and heterogeneous nature. Specifically, FL suffers from long transmission latency to a remote cloud server [4]. Additionally, the varying qualities of each MD's dataset can impact the accuracy of the global model, and the local training burden may make MDs reluctant to participate in the FL process [5–7].

To tackle these challenges, researchers have explored the concept of mobile edge computing (MEC)-assisted FL, which leverages the computing capabilities available at the mobile network edge to facilitate intermediate aggregation and streamline the FL process effectively [8]. The objective is to achieve optimal FL operations. In particular, there have been active investigations into efficient FL participant selection, aggregation management, and radio and computation resource management (such as CPU/GPU). In terms of radio and computation resource management, given the constraints of limited communication bandwidth and computational capacity, various approaches to joint edge association and the allocation of radio and computation resources have been explored. These endeavors aim to enhance the accuracy of the global model or improve the speed of convergence as well as energy efficiency [6,9–19].

However, most existing research assumes that participating MDs fully utilize their given datasets, considering the fact that more data generally leads to higher accuracy in DL. In practice, however, MDs may have limited computing and battery capabilities, making it burdensome for MDs if they have large datasets. The burden placed on MDs due to the full utilization of their datasets in the FL process often leads to a reduction in their participation, resulting in a decrease in global accuracy. Thus, this issue requires careful management. Nevertheless, current studies tend to focus on providing incentives to promote MD participation rather than addressing the burden on MDs [20–22]. Only a few studies have considered dataset management for efficient data utilization. For instance, Duan et al. [7] proposed the Astrea framework, which adaptively conducts data augmentation and downsampling to address local data imbalance. Similarly, Ren et al. [9] proposed a joint batch size selection and communication resource allocation algorithm for both CPU and GPU resources, taking into account the trade-off between loss decay and latency. Additionally, Kim et al. [6] proposed an energy-efficient joint dataset and computation management scheme that utilizes dataset management to reduce the burden on MDs, enabling energy-efficient FL with minimal accuracy loss. However, most of these works assume that the value of the dataset remains constant throughout the FL process. In non-independent and identically distributed (non-IID) cases, both the amount of data and the class distribution across rounds can impact the global accuracy as mentioned in [23,24]. To the best of our knowledge, the careful consideration of dataset management for alleviating the burden on MEC-assisted FL processes with minimal accuracy loss, specifically addressing the diminishing effect of dataset size over rounds and class distribution, has not been thoroughly explored in previous studies. This constraint highlights an area that requires further attention and research.

In this article, we propose an adaptive dataset management (ADM) scheme to alleviate the burden on MDs in the FL process while minimizing accuracy loss. The detailed contributions of this article are as follows:

- Considering the diminishing effect of dataset size over rounds, as validated by our empirical study, along with the distribution of classes, we design the proposed ADM scheme. This scheme incorporates both the data size adjustment algorithm and the class adjustment algorithm.
- To develop the proposed ADM scheme, we present rigorous analytical models to estimate accuracy and end-to-end service latency concerning the dataset size, taking into account the proposed discount factor. Subsequently, to balance between the estimated accuracy and end-to-end service latency, we formulate the objective function based on the ratio of these two factors.
- Regarding the dataset size adjustment problem, we determine the optimal dataset size adjustment across clients by solving an optimization problem, initially a non-convex problem due to the presence of a non-differentiable function. We employ several mathematical techniques to transform it into a convex optimization problem and provide the global optimum solution.
- In addressing the class adjustment problem, we establish the optimal dataset size adjustment across classes for each client, considering the class distribution over MDs

in the non-IID case. Here, we propose a greedy-based heuristic algorithm to reduce the Kullback–Leibler Divergence (KLD) distance and derive a suboptimal solution with low complexity.
- As a practical consideration, we also provide a detailed discussion on the implementation of the proposed ADM on a virtualization platform, along with a prototype of the proposed framework. Finally, simulation results demonstrate the effectiveness of the proposed scheme in reducing the training burden on MDs while maintaining acceptable training accuracy.

The remainder of this paper is organized as follows. Section 2 summarizes the previous works related to resource management for MEC-assisted FL. Section 3 provides the proposed system model of the ADM scheme. Section 4 provides our proposed ADM scheme, which consists of (i) dataset size adjustment and (ii) class adjustment and detailed discussions on its implementation. In Section 5 evaluates the performance of the proposed system. Finally, Section 6 concludes the paper.

2. Related Work

Research on MEC-assisted FL has seen numerous approaches in the literature aimed at achieving optimal dataset management of FL. It is important to note that effective data management is essential for achieving optimal performance in Internet of Things (IoT) systems. In particular, as FL has been extensively utilized in IoT systems, there is a growing interest in addressing this issue to enhance FL performance [25]. In this brief survey, we will specifically focus on recent advancements in dataset management for MEC-assisted FL. The work by [9] focused on accelerating the training process in CPU and GPU computation scenarios. They proposed a joint batch size selection and radio resource allocation algorithm. This approach aimed to optimize the training process by effectively utilizing both CPU and GPU resources. In [6], the authors developed an analytical model to optimize learning efficiency and energy consumption. They proposed a joint dataset and computation management approach to strike a balance between these two factors. In [26], the authors proposed an adaptive batch size and learning rate selection algorithm. Their objective was to mitigate the negative impact of the synchronization barrier in FL. The authors of [27] proposed a detailed data selection method to enhance learning efficiency by choosing only the data that contributes to the improvement of the model's performance. However, these studies did not take into account the distribution of data. Other research has focused on considering the distribution of data, which can be classified into data adjustment and client selection/scheduling approaches. In [7], the authors proposed a self-balancing algorithm to alleviate data imbalance. This was achieved by augmenting the minority class and downsampling the majority class on MDs. Additionally, they proposed a mediator-based rescheduling algorithm to select MDs and distribute aggregated data to achieve near-IID distribution. In [28], a data augmentation strategy using a generative adversarial network (GAN) was employed. This strategy enabled each device to locally reproduce the data samples of all MDs, therefore promoting an IID distribution of data. The authors of [29] introduced FedSwap, a method that swaps models among the MDs in each round to alleviate data diversity. This approach aimed to achieve a more consistent data distribution across devices. In [30], the authors proposed the CSFedAvg algorithm, which selects clients with a low degree of non-IID data by utilizing weight divergence. Similarly, The authors of [31] employed DRL to identify clients relevant to the application task, ensuring they have uniform data across a large dataset. These strategies are designed to enhance the aggregation process by focusing on clients with similar data distributions. However, these approaches, which focus on aligning IID data through data augmentation or client selection, tend to overlook the learning efficiency of MDs. We propose ADM, taking into consideration the aforementioned limitations. ADM accelerates the learning process by adjusting the dataset in response to diminishing the effect of dataset size over rounds while also adjusting the class distribution by considering non-IID data of MDs. We present a list summary of related works in Table 1.

Table 1. Summary of related works.

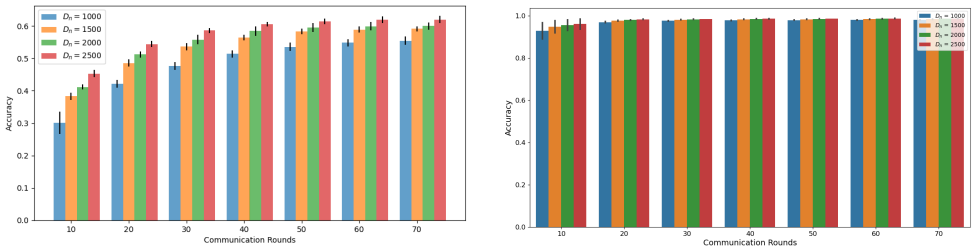
Related Works	Topic	Key Contributions
[6]	Learning Efficiency	Propose dataset and computation management strategy
[7]	Data Distribution Management	Augment the minority class and downsample the majority class
[9]	Learning Efficiency	Propose batch size selection and radio resource allocation algorithm
[26]	Learning Efficiency	Propose an adaptive batch size and learning rate selection algorithm
[27]	Learning Efficiency	Choose the data contributing to the improvement of the model
[28]	Data Distribution Management	Propose data augmentation strategy using GAN to promote IID data
[29]	Data Distribution Management	Swap models among the MDs to alleviate data distribution
[30]	Data Distribution Management	Select clients with a low degree of non-IID data using weight divergence
[31]	Data Distribution Management	Select clients relevant to the application task employing DRL
ADM	Learning Efficiency Data Distribution Management	Balance accuracy and latency by adjusting dataset size Propose a method for diminishing the effect of dataset size over rounds Adjust class distribution on non-IID data

3. System Model

3.1. Motivating Example for Discounting Factor

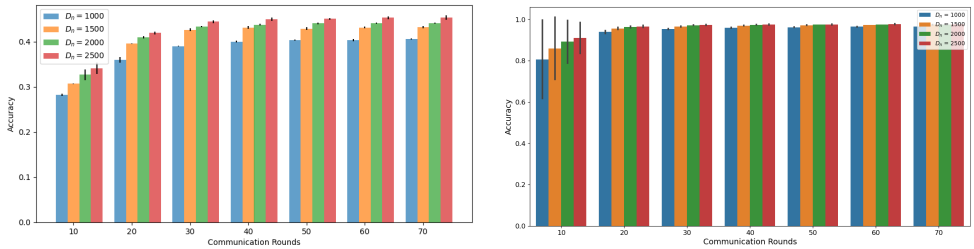
It is well-established that the training accuracy of FL generally improves as the dataset size per MD increases. However, the rate at which the accuracy increment, also known as accuracy gain, tends to decrease as the communication rounds progress [6,18]. To further investigate the impact of dataset size on accuracy improvement over communication rounds, we conducted an empirical study, which led us to introduce a discount factor in our proposed scheme.

Our simulations, as depicted in Figures 1 and 2, demonstrate the accuracy trend of the global model in the FL framework when using the CIFAR-10 and MNIST datasets. Under both the Independent and Identically Distributed (IID) and non-IID settings, we observed that the accuracy increases with an increasing amount of dataset denoted as D_n, where n represents the index of MDs at a specific communication round, as confirmed by previous studies [6,18]. However, in this experiment, where the total number of MDs is 10, we noted that the accuracy improvement from utilizing a larger dataset size, D_n, diminishes as the communication rounds progress. Notably, the gap between $D_n = 1000$ and $D_n = 2500$ decreases consistently, supporting this observation. This trend becomes more evident when we analyze the accuracy gain based on communication rounds with respect to D_n, as illustrated in Figures 3 and 4. These demonstrate that a significant accuracy gain occurs in the initial rounds, but after 40 rounds, the accuracy gain significantly decreases. The high variance observed in the initial rounds reflects greater volatility in accuracy gain compared to other rounds. This indicates a tendency for accuracy to increase significantly in the early rounds. Based on these findings, we can conclude that to alleviate the training burden on MDs, it is advisable to reduce the dataset size more aggressively as the communication rounds evolve. To account for this trend in adaptive data management, we propose the introduction of a discounting factor that considers the accuracy gained from the dataset in relation to the communication rounds.



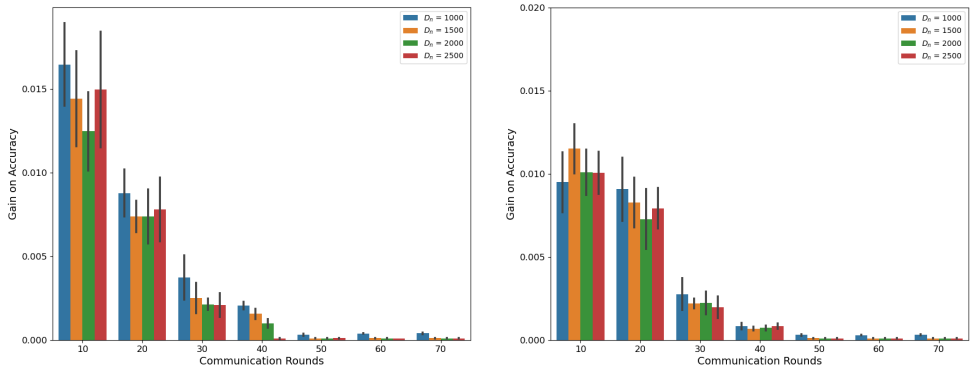
(**a**) (**b**)

Figure 1. Accuracy based on communications rounds and D_n on IID setting (**a**) CIFAR-10 (**b**) MNIST.



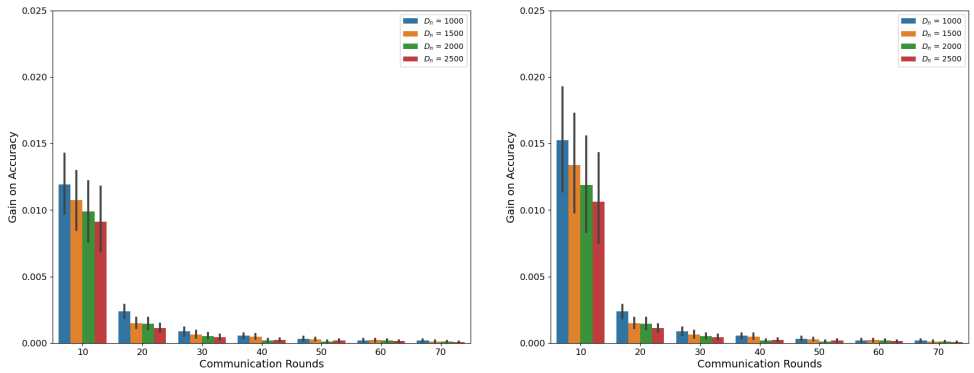
(**a**) (**b**)

Figure 2. Accuracy based on communications rounds and D_n on Non-IID setting (**a**) CIFAR-10 (**b**) MNIST.



(**a**) (**b**)

Figure 3. Accuracy gain based on communications rounds and D_n on CIFAR-10 (**a**) IID (**b**) Non-IID.



(**a**) (**b**)

Figure 4. Accuracy gain based on communications rounds and D_n on MNIST (**a**) IID (**b**) Non-IID.

3.2. Proposed System Model

For ease of reference, we present Table 2, which comprises a list of the key symbols that we define and utilize in this paper.

Table 2. Summary of Major Symbols.

Symbol	Definition
$\mathbf{N}$	Set of MDs
N	Total number of MDs
$\mathbf{D_n}$	Dataset of MD n
D_n	Total number of samples in MD n's dataset
D'_n	Adjusted number of samples in MD n's dataset
σ	Discounting factor
A	Accuracy estimation model
c_n	Number of CPU cycle of MD n required to process one samples
f_n	CPU frequency of MD n
L_n^c	Local model computation latency
I_n	Number of training local model iteration
B	Available bandwidth at MEC server
R_n	Transmission rate of MD n
W_n	Size of the local model parameters of MD n
L	End-to-end service latency of FL
v_n	Dataset adjustment vector
D_{KL}	Kullback–Leibler divergence distance
$v_{n,k}$	Class adjustment vector

As depicted in Figure 5, the proposed system architecture supports an FL framework consisting of multiple MDs and a single MEC server where the MEC server is directly connected to a single base station (BS) serving MDs. Here, we define $\mathbf{N}$ as the set of MDs, where $|\mathbf{N}| = N$ denotes the total number of MDs. For each $n \in \mathbf{N}$, MD n has local dataset $\mathbf{D_n} = ((x_n^1, y_n^1), (x_n^2, y_n^2), (x_n^3, y_n^3), \ldots, (x_n^i, y_n^i), \ldots, (x_n^{|\mathbf{D_n}|}, y_n^{|\mathbf{D_n}|}))$, where $|\mathbf{D_n}|=D_n$ denotes the total number of samples, x_n^i and y_n^i is the i-th data sample and corresponding ground-truth label, respectively.

In this context, the MEC server, acting as a centralized server, orchestrates the coordination of FL tasks across multiple MDs. Each MD performs local training on its local dataset and sends the local update to the MEC server. Subsequently, the MEC server aggregates the local updates from the MDs to generate the global model. Throughout this process, the MEC server employs an adaptive data management (ADM) scheme to adjust the dataset size assigned to each MD, therefore achieving the lightweight FL process. The entire procedure can be categorized into the following three steps:

- Step 1: In Step 1, the MEC server selects appropriate MDs as FL participants. Then, the MEC server requests and receives the class distribution of the dataset from each MD to conduct the ADM scheme, which will be explained in detail in Section IV. Using the information obtained from the selected MDs, the data adjustment message is calculated using the ADM scheme. Afterward, the MEC server initiates the task by providing an initial shared global model, denoted as w_G^0, and the data adjustment message for local training to multiple MDs. The initial shared global model may include a TensorFlow graph, weights, and instructions.

- Step 2: Each MD n conducts local training on the adjusted dataset D'_n among entire local data D_n using the shared global model (i.e., w_G^0 in the initial round or w_G^t in round t). Specifically, by minimizing the loss function $\mathcal{L}(w_n)$, the local model parameter w_n at MD n is given by

$$w_n^* = \arg\min_{w_n} \mathcal{L}(w_n). \tag{1}$$

  Then, the updates are transferred to the MEC server.
- Step 3: The MEC server combines the local model updates from the MDs and generates a global model by solving an optimization problem that minimizes the global loss function.

$$\mathcal{L}(w_G^t) = \frac{1}{N} \sum_{i \in I} \mathcal{L}(w_i). \tag{2}$$

  Then, the MEC server sends the updated global model parameters back to the MDs.

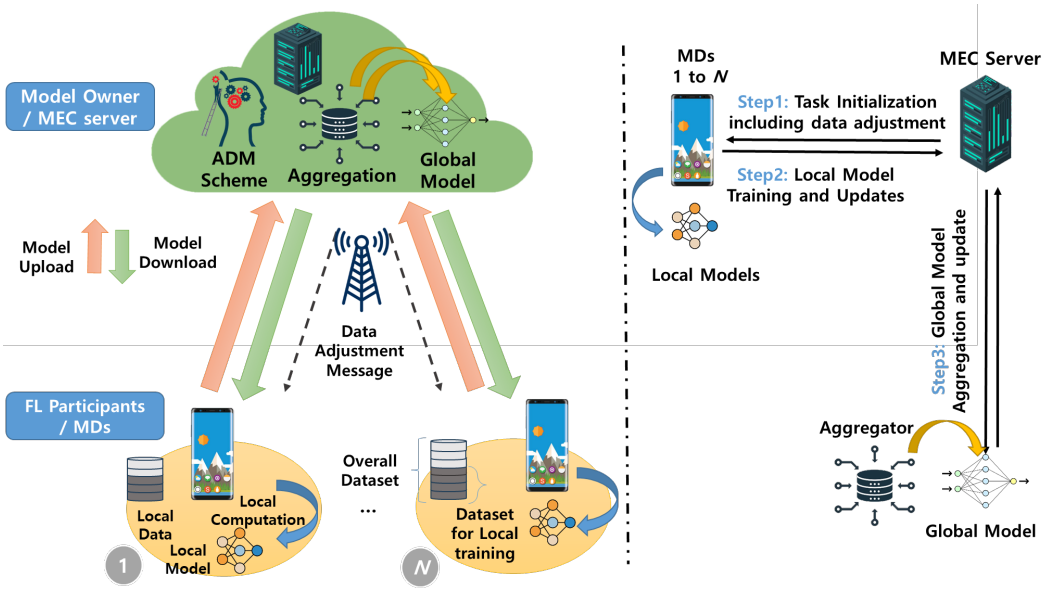


Figure 5. Proposed system architecture.

Multiple rounds of the FL process, which include Steps 2–3, are iterated until either the global loss function at the MEC server converges to the termination condition or the specified target accuracy is achieved. It should be noted that the FL process allows for the selection of different types of ML models depending on the specific application of the FL service. Additionally, in Step 3, the aggregation of the global model, an essential component of FL, can be accomplished using various mechanisms such as the federated averaging algorithm and secure aggregation algorithm [32].

3.3. Analytical Models

In this subsection, we represent analytical models by formulating (i) an accuracy estimation model and (ii) an end-to-end service latency model, respectively.

3.3.1. Accuracy Estimation Model

In most of the literature [6,18], the estimated training accuracy in FL can be modeled as either a concave or linear function, depending on the observation range. The concave behavior of the accuracy can be approximated by piecewise linear approximation. To simplify our proposed scheme and control the dataset size within specific ranges, as shown in [6], we adopt a linear function in this paper. Furthermore, considering the impact of dataset size on accuracy improvement over communication rounds, we introduce a discounting factor (a number between 0–1), which provides a clever way to scale down the

impact of accuracy improvement increasingly after each round evolves. Then, the proposed accuracy estimation model A is given by

$$A = \sigma^{r-1} \sum_{n \in \mathbf{N}} D_n, \tag{3}$$

where σ is the discounting factor and r is a communication round index. It represents the extent to which the accuracy improvement is discounted with respect to the dataset size of MDs over communication rounds. If $\sigma_n = 1$, the accuracy improvement remains sustained regardless of the communication rounds. However, when $\sigma_n < 1$, the impact of the dataset size on accuracy improvement gradually diminishes as the number of communication rounds increases.

3.3.2. End-to-End Service Latency Model

Following [6], the end-to-end service latency comprises two components: (i) computation latency and (ii) transmission latency between MD n and the MEC server. The computation latency includes the MD n's local model computation latency, denoted as L_n^c, and the MEC server's computation latency for global model aggregation. Considering that global model aggregation is typically a lighter task compared to local model training, we assume that the latency associated with global model aggregation is negligible. This assumption aligns with various studies [6] that assume the MEC server possesses sufficient resources to handle this task. Let c_n be the number of CPU cycles of MD n required to process one sample, and f_n is the CPU frequency. Then, the local model computation latency, L_n^c, can be expressed as

$$L_n^c = I_n \frac{c_n D_n}{f_n}, \tag{4}$$

where the I_n denotes the number of training local model iterations.

The transmission latency, denoted as L_n^t, encompasses both the uploading of the local model and the downloading of the global model. However, since the latency associated with downloading the global model is negligible compared to uploading the local model, we can disregard it in our analysis, as assumed in [6]. To formulate the transmission latency L_n^t, we consider that the MEC server fairly assigns bandwidth to all MDs, with an available bandwidth denoted as B. The transmission rate of MD n can be defined as:

$$R_n = \frac{B}{N} \ln\left(1 + \frac{h_n p_n}{N_0}\right), \tag{5}$$

where h_n, p_n, and N_0 represent the channel gain, transmission power of MD n, and the background noise, respectively. The transmission latency for the local model parameters w_n from MD n to the MEC server can be formulated as:

$$L_n^t = \frac{W_n}{R_n}, \tag{6}$$

where W_n denotes the size of the local model parameters of MD n.

Finally, the end-to-end service latency of each MD n is defined as $L_n = L_n^c + L_n^t$. In FL, the global model aggregation is conducted when the MEC server receives all local models from all MDs, so the total end-to-end service latency of FL, which is determined by the slowest MD, is defined as

$$L = \max_{n \in \mathbf{N}} \{L_n^c + L_n^t\}. \tag{7}$$

4. Proposed ADM Scheme

In this section, we present a novel adaptive dataset management (ADM) scheme for lightweight FL frameworks. The goal of this scheme is to determine the optimal dataset adjustment vector v_n, which represents the ratio between the amount of local dataset used for local training and the total amount of local dataset. Additionally, we address the

problem of determining which class of data should be reduced, taking into account the class distribution.

4.1. Dataset Size Adjustment

To strike a balance between accuracy improvement and end-to-end service latency with respect to the dataset size, we define an optimization problem aimed at maximizing the ratio of accuracy estimation A to end-to-end service latency L in FL. In this formulation, we replace D_n with the adjusted dataset D'_n, where $D'_n = v_n D_n$. Then, the problem can be formulated as follows:

Prob.1 :

$$\max_{v_n} \quad \frac{A}{L} \tag{8a}$$

$$\text{s.t.} \quad \gamma_d \leq v_n \leq 1, \forall n \in \mathbf{N}, \tag{8b}$$

where the constraint (8b) specifies the range of v_n with lower bound parameter γ_d for the selected MEC MDs.

To convert *Prob*.1 into standard minimization form by plugging (3)–(7) into *Prob*.1, the *Prob*.1 is newly defined as follows:

Prob.2:

$$\min_{v_n} \quad -\sigma^{r-1} \frac{\sum_{n\in\mathbf{N}} v_n D_n}{\max_{n\in\mathbf{N}} \{I_n \frac{c_n v_n D_n}{f_n} + \frac{W_n}{R_n}\}} \tag{9a}$$

$$\text{s.t.} \quad \gamma_d \leq v_n \leq 1, \forall n \in \mathbf{N}, \tag{9b}$$

However, since *Prob*.2 has the form $\max(.)$ in the objective function, which is not differentiable, it is a non-convex optimization problem. Thus, we convert $\max(.)$ into an affine function by introducing an additional variable t and letting

$$t = \max_{n\in\mathbf{N}} I_n \frac{c_n v_n D_n}{f_n} + \frac{W_n}{R_n}. \tag{10}$$

This new variable t induces an additional constraint:

$$I_n \frac{c_n v_n D_n}{f_n} + \frac{W_n}{r_n} \leq t, \forall n \in \mathbf{N}. \tag{11}$$

Using this additional constraint with the new variable t, we can rewrite *Prob*.2, which is a problem equivalent to

Prob.3:

$$\min_{v_n, t} \quad -\sigma^{r-1} \frac{\sum_{n\in\mathbf{N}} v_n D_n}{t} \tag{12a}$$

$$\text{s.t.} \quad \gamma_d \leq v_n \leq 1, \forall n \in \mathbf{N}, \tag{12b}$$

$$I_n \frac{c_n v_n D_n}{f_n} + \frac{W_n}{R_n} \leq t, \forall n \in \mathbf{N}. \tag{12c}$$

Lemma 1. *Prob. 3 is a linear programming (LP) problem with respect to optimization variables (v_n).*

Proof. First, the objective function is linear with respect to v_n. Moreover, the inequality constraints (12b) and (12c) are affine in terms of the optimization variables (v_n). Consequently, since both the objective function and the inequality constraints are affine, the problem is an LP problem in terms of the optimization variables (v_n). □

Lemma 2. *The Prob. 3 is a strictly increasing function with respect to optimization variable (t), and t^* should be at the lower bound of constraint (12c).*

Proof. As *Prob*.3 includes the term $-\frac{1}{t}$ in the objective function, it is evident that the objective function is strictly increasing with respect to t. Consequently, the optimal value of t (t^*) resides at the lower bound of (12c). □

Lemmas 1 and 2 form the basis for solving *Prob*.3 using the block coordinate descent method [33]. In this method, given a fixed value of t, we can readily solve for the optimal values of v_n using the Simplex algorithm (SA). Subsequently, t^* and (v_n) are obtained iteratively by mutually fixing each other until the cost function converges, following the block coordinate descent approach. The algorithmic procedure is summarized in Algorithm 1.

Lemma 3. *Algorithm 1 performs a sublinear convergence rate.*

Proof. The block coordinate descent method demonstrates a sublinear convergence rate, as proved in [34]. Based on Lemmas 1 and 2, the proposed algorithm is designed using the block coordinate descent method. Consequently, Algorithm 1 has a sublinear convergence rate. □

Algorithm 1 ADM scheme—data size adjustment

Input : $D_n, r, \sigma, W_n, c_n, f_n, I_n, R_n, \theta_c$
Initialize : v_n and t are randomly initialized within the constraints.
Output : Optimal $v_n^* D_n = D_n'^*$

```
while True do
    v_n ←solving Prob.3 via SA
    t ←max_{n∈N} I_n c_n v_n D_n / f_n + W_n / R_n
    C_i ←(12a)
    if |C_i − C_{i−1}| < θ_c then
        break
    end if
    i = i + 1
end while
v_n^* ← v_n
D_n'^* ← ⌊v_n^* D_n⌋
```

4.2. Class Adjustment

After obtaining D_n' from the previous Algorithm 1, under the assumption of an IID case, it is obvious that all MDs can reduce their entire class of dataset evenly to achieve $D_n'^*$ which is smaller than D_n. However, in non-IID cases, we should carefully determine which class of data should be reduced, taking into account the class distribution. Specifically, during data reduction, the proposed ADM aims to achieve a uniform distribution of a class of entire datasets across MDs, known as an IID. This approach minimizes the accuracy loss resulting from data adjustment while reducing the training burden on MDs. Here, we define $\mathbf{K}$ as the set of classes in the dataset, where $|\mathbf{K}| = K$ denotes the total number of classes. To accomplish this, we utilize the Kullback–Leibler divergence (KLD) distance to quantify the proximity between the class distribution of the dataset and the uniform distribution, which is formulated by

$$D_{KL}(P_{\mathbf{N}}||P_u) = \sum_{k \in \mathbf{K}} P_N(k) \ln\left(\frac{P_N(k)}{P_u}\right), \tag{13}$$

where the P_u is the uniform distribution and $P_{\mathbf{N}}$ is the class distribution of the entire dataset from all MDs.

For each $n \in \mathbf{N}$, consider that local dataset of MD n $\mathbf{D_n}$ has disjoint subset $\mathbf{D_{n,k}}$, which satisfies

$$\mathbf{D_n} = \bigcup_{k \in \mathbf{K}} \mathbf{D_{n,k}}, \tag{14}$$

where $\mathbf{K}$ is the set of class in the dataset, and $|\mathbf{D_{n,k}}| = D_{n,k}$ denotes the total number of samples of class k in the MD n's dataset.

As depicted in Figure 6, the overall procedure can be categorized into the following three steps:

- Step 1: Each MD n calculate and send their class distribution vector $[D_{n,1}, D_{n,2}, \ldots, D_{n,K}]$ to the MEC server, where $D_{n,k}$ is the number of the samples with label k of MD n.
- Step 2: The MEC server executes the proposed ADM scheme. As step 2-1, using Algorithm 1, dataset size adjustment $D_n'^*$ including v_n^* is determined. After that, to minimize the $D_{KL}(P_{\mathbf{N}}||P_u)$ for making the class distribution of aggregated dataset close to the IID, as step 2-2, class adjustment is conducted. By aggregating such class distribution vector over MDs, the MEC server can calculate the adjustment of data for each class, which is given by

$$reduction_{n,k} = D_{n,k} - reduction_{level}, \forall n \in \mathbf{N}, \forall k \in \mathbf{K}. \tag{15}$$

  where $reduction_{n,k}$ represents the reduction applied to each class dataset of MD n, and $reduction_{level}$ denotes the target dataset size to be retained across all classes. In this scenario, if (15) yields a value less than 0, it implies that the class size is already below the target dataset size, $reduction_{level}$. Thus, $reduction_{n,k}$ should be set to 0. The process continues until the condition $\sum_{k=1}^{K} D_{n,k} - reduction_{n,k} = v_n^* \cdot D_n$ is satisfied. During this process, $reduction_{level}$ is gradually decreased by subtracting a constant Δ iteratively. Finally, the optimal class distribution vector is updated according to the equation:

$$D_{n,k}^* = \lceil D_{n,k} - reduction_{n,k} \rceil, \forall n \in \mathbf{N}, \forall k \in \mathbf{K}. \tag{16}$$

- Step 3: Finally, as step 3, updated class distribution vector $[D_{n,1}^*, D_{n,2}^*, \ldots, D_{n,K}^*]$ is delivered to the MDs.

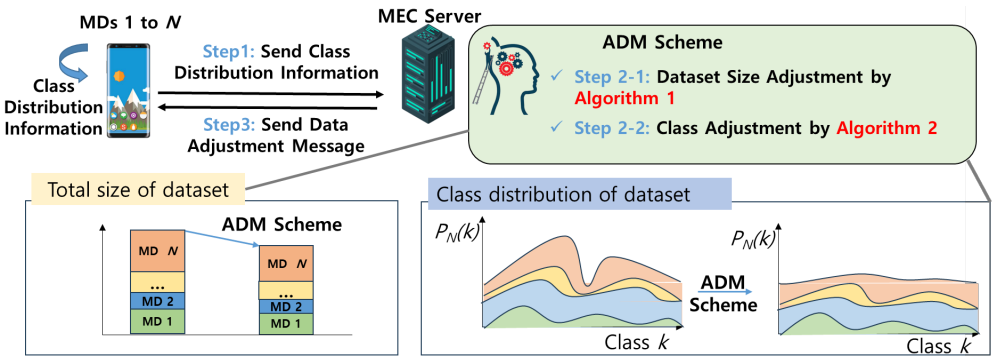


Figure 6. The overall process of the ADM scheme.

The class adjustment algorithm is summarized in Algorithm 2. Here, it is obvious that the time complexity is O(NK), which is simple and easily deployable in the real world with the limited number of MDs and classes, where N is the number of MDs and K is the number of unique labels.

Algorithm 2 ADM scheme—class adjustment

Input : $[D_{n,1}, D_{n,2}, \ldots, D_{n,K}]$, v_n^*
Output : Optimal $D_{n,k}^*$

for each MD $n \in \mathbf{N}$ **do**
  $D_{max} = \max\{D_{n,1}, D_{n,2}, \ldots, D_{n,K}\}$
  $reduction_{level} = D_{max} - \Delta$
  while $\sum_{k=1}^{K} D_{n,k} - reduction_{n,k} \neq v_n^* \cdot D_n$ **do**
    $$reduction_{n,k} = \begin{cases} D_{n,k} - reduction_{level} & \text{, if } D_{n,k} - reduction_{level} \geq 0 \\ 0 & \text{, otherwise} \end{cases}, k \in \mathbf{K}$$
    $reduction_{level} = reduction_{level} - \Delta$
  end while
  for $k \in \mathbf{K}$ **do**
    $D_{n,k}^* = \lceil D_{n,k} - reduction_{n,k} \rceil$
  end for
end for

4.3. Discussion on Implementation of the Proposed ADM on Virtualization Platform: Kubernetes

For practical considerations, this subsection provides a discussion on the implementation of the proposed ADM on Docker-based Kubernetes platforms. As in our previous study [35], the proposed ADM can easily be implemented on a Kubernetes platform consisting of a master and several nodes.

Figure 7 illustrates the implementation architecture of the proposed ADM on the Kubernetes platform. In this example scenario, as depicted in Figure 7, we consider that the architecture comprises one master and two nodes. Specifically, the master is responsible for managing the two nodes using Kubernetes and aggregating the global model for FL. The nodes, on the other hand, are responsible for running pods. Within each pod, there is a container that contains the FL framework received from the master. Consequently, the pod performs local training for the FL by executing the FL framework within the container. To incorporate the ADM scheme into this platform, we also developed the necessary signaling messages for data adjustment (https://github.com/juneseokBang/FL_K8S, accessed on 1 March 2024).

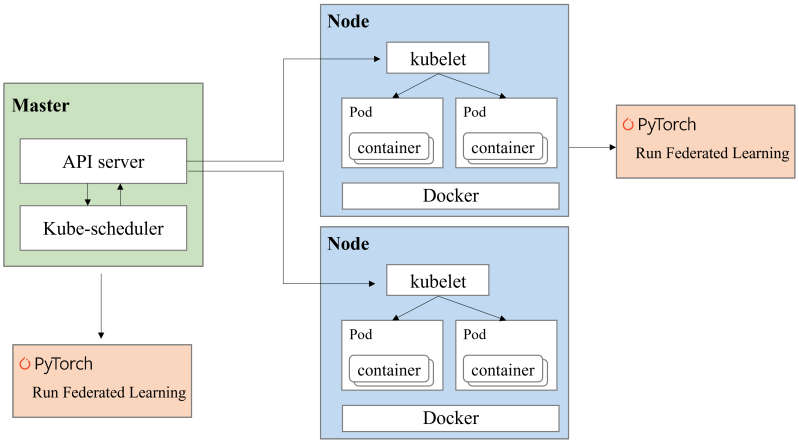


Figure 7. Implementation of the proposed ADM on a virtualization platform.

Furthermore, we also implemented a GUI platform that allows users to select FL parameters and compare the trained results using graphical representations (https://github.com/juneseokBang/FL_GUI, accessed on 1 March 2024). Figure 8 illustrates the workflow of the GUI platform. When a user selects a parameter on the web interface, the value is transmitted to the FL server (as shown in Figure 9a,b). The server performs FL using the received parameter value. Once the training is completed, the web interface displays the learning results graphically (as depicted in Figure 9c). Additionally, the platform retains a

history of past training results, enabling easy graph comparisons. Although FedAvg is now implemented as a basic algorithm, we can add new parameters or new algorithms to the platform, ensuring its scalability and adaptability to evolving research requirements. This capability empowers researchers to explore a wider range of experimental configurations and evaluate their impact on model performance easily.

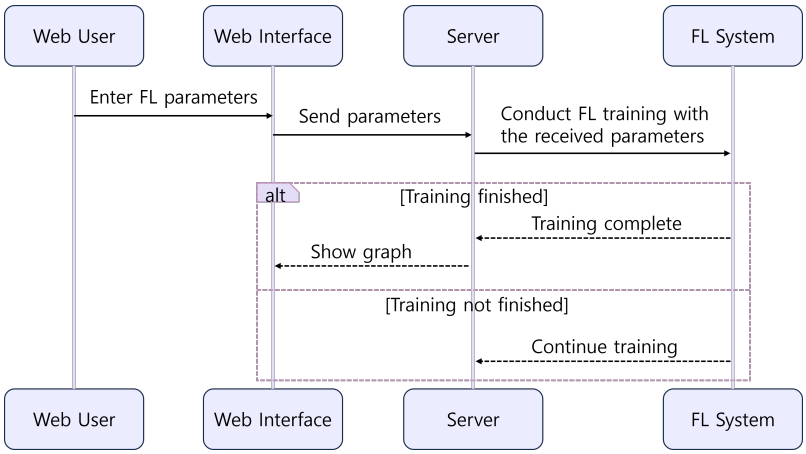


Figure 8. GUI Platform Workflow.

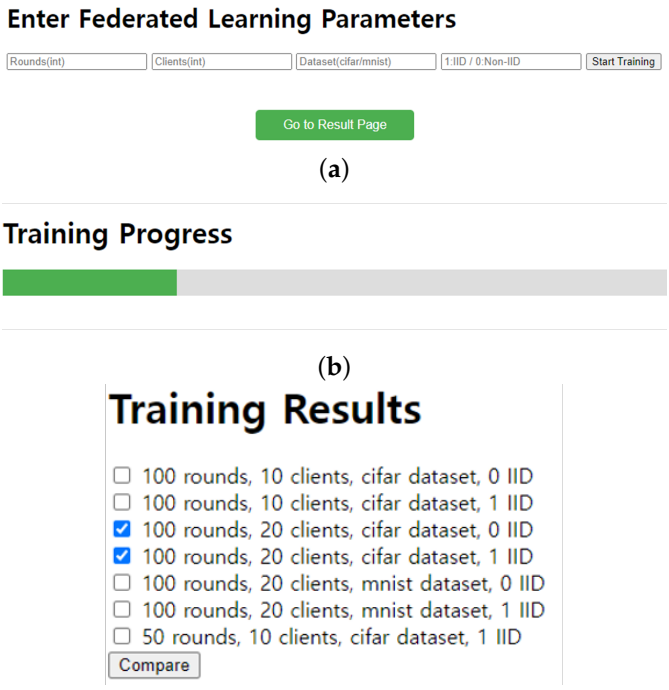
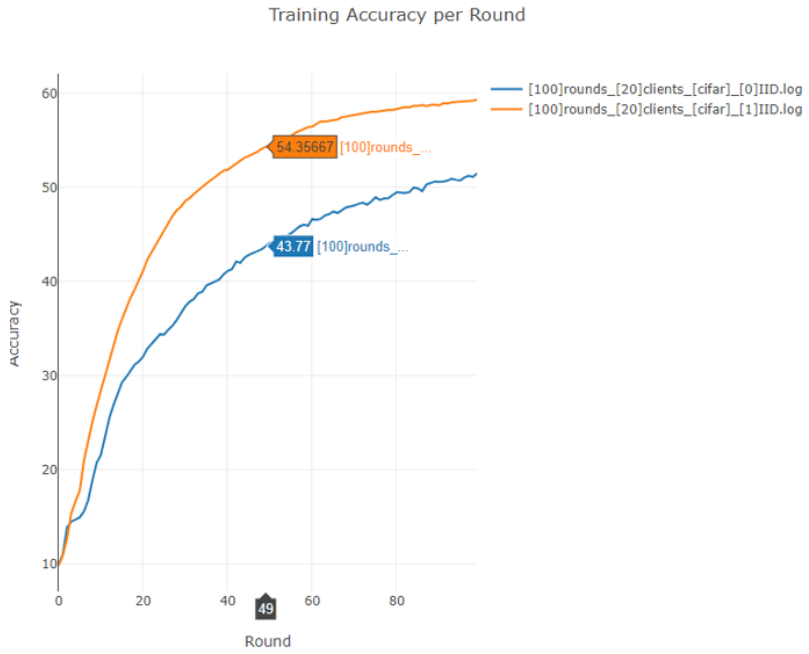


Figure 9. *Cont.*



(**c**)

Figure 9. GUI platform. (**a**) entry of FL hyperparameters (**b**) training progress (**c**) training result comparison.

5. Performance Evaluation

In this section, we present simulation results to validate the effectiveness of the proposed ADM scheme compared to two benchmarks. Benchmark 1 (B1) represents FedAvg applied without any dataset management method, utilizing all training data [1]. Benchmark 2 (B2) represents FedAvg with heuristic dataset management (HDM) [36]. HDM employs a strategy to reduce the size of the training dataset by 20% every 20 rounds without accounting for unbalanced class distribution. We evaluated ADM by training popular CNN models on two datasets. (1) MNIST, a dataset that has 60 K 28 $\times$ 28 training images of 10 classes; (2) CIFAR-10, a dataset that contains 50 K 32 $\times$ 32 colored images of 10 classes. In the simulated environment, we assume that the computational capacity f_i of each MD is 3 GHz, which is fairly assigned to all MDs. Other simulation parameters and hyperparameters are listed in Table 3.

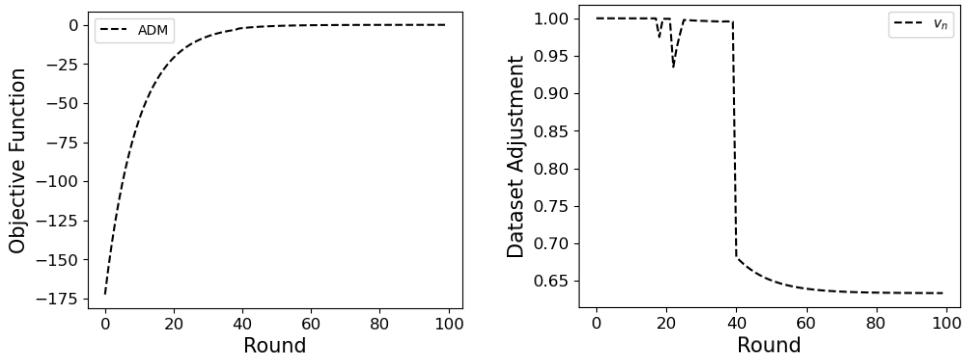
Table 3. Simulation Parameters.

Parameter	Value
Number of CPU cycles (c_n)	30 cycles/sample
Computation capacity (f_n)	3 GHz
Noise power (N_0)	-114 dBm
Size of local model (W_n)	100 Kbits
Gamma (γ_d)	0.4
Number of MDs (N)	20
Discounting factor (σ)	0.9 $\times 10^{-8}$

5.1. Numerical Analysis—Dataset Size Adjustment

In this subsection, we demonstrate how ADM adjusts dataset size considering **Prob.3** (12a). We assume all MDs have 2500 local datasets and IID data. As shown in Figure 10a,b, v_n decreases when the objective function converges (near round 40). In Figure 10a, the objective function converges to zero as the round passes due to discounting factor σ in (3). Near round 40, v_n drops the dataset to 65% because the influence of the dataset on

the accuracy gain significantly diminishes. Consequently, not all datasets are required to update the global model.



(**a**) (**b**)

Figure 10. Objective function (**a**) and the average dataset adjustment v_n (**b**) over rounds.

5.2. Simulation Analysis—Dataset Size Adjustment: IID Case

We assume the initial number of training data samples for each MD is 2500 for CIFAR-10 and 3000 for MNIST (i.e., $|\mathbf{D_n}|$ = 2500 and 3000, respectively), and N is 20, with the data being IID. Table 4 shows the accuracy of ADM and two benchmarks. Data management methods (HDM and ADM) ensure accuracy while reducing the training data, highlighting a significant diminishing effect of dataset size as the rounds progress. However, when the number of MDs is large, and the dataset size is small, the performance of HDM significantly decreases. For CIFAR10, the accuracy of HDM is 50.54%, which is a 6.19% decrease compared to FedAvg. In contrast, ADM achieves an accuracy of 53.73%, demonstrating robustness in scenarios with small datasets. The reduction in HDM's performance is attributed to its lack of consideration for dataset size, whereas ADM adaptively manages the dataset by taking into account the dataset size of each MD.

Table 4. Test Accuracy.

Method	Data Distribution	MNIST (N = 20, D_n = 3000)	CIFAR-10 (N = 20, D_n = 2500)	MNIST (N = 50, D_n = 1250)	CIFAR-10 (N = 50, D_n = 1000)
FedAvg (B1)	IID	99.32%	54.80%	98.83%	53.88%
	Non-IID (β = 0.8)	98.98%	49.63%	98.48%	49.54%
HDM (B2)	IID	99.20%	54.20%	98.54%	50.54%
	Non-IID (β = 0.8)	98.86%	49.59%	98.04%	48.53%
ADM	IID	99.25%	54.65%	98.68%	53.73%
	Non-IID (β = 0.8)	98.92%	49.54%	98.35%	49.25%

As shown in Table 5, when training with all datasets (i.e., B1) for MNIST, it takes about 40.19 s per round, whereas ADM reduces this to approximately 26.88 s per round, representing a 33.1% reduction. Similarly, for CIFAR-10, the time decreases from 35.61 s to 26.71 s, marking a 24.99% reduction. This significantly accelerates the training process. Although HDM reduces the training time more than ADM does, ADM achieves an accuracy comparable to FedAvg (B1), whereas HDM does not guarantee this accuracy. Figure 11 illustrates how ADM adjusts data considering the dataset size. We conducted a total of 100 rounds and averaged every 10 rounds to represent the mean value of vn in each round. If each MD has 1000 data samples, the data reduction rate is not significantly reduced as rounds progress, whereas, with 3000 data samples, the reduction rate drops to 60%. This indicates that the v_n is influenced by the dataset size, as outlined in (12a). For D_n = 3000 and D_n = 2000, volatility increases notably in the 30 s and 40 s rounds, respectively. This

suggests that these points are where significant adjustments to v_n occur. Therefore, ADM maintains the performance of the local model even with smaller datasets.

Table 5. Average Training Time when N = 50 on IID case.

Second/Round (Average)	MNIST (D_n = 1250)	CIFAR-10 (D_n = 1000)
FedAvg (B1)	40.19 s	35.61 s
HDM (B2)	24.34 s	22.14 s
ADM	26.88 s	26.71 s

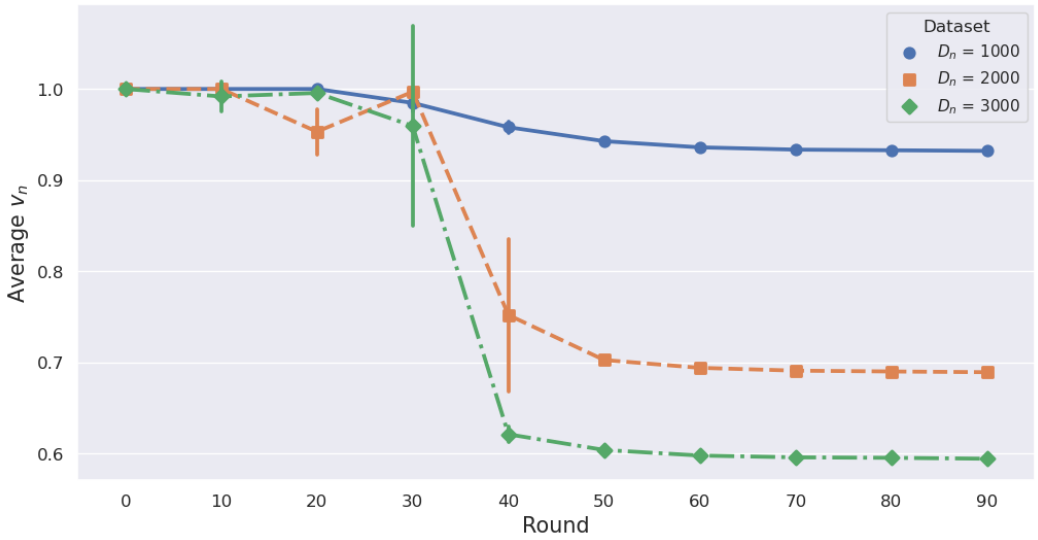


Figure 11. Average Data Reduction Rate per 10 Rounds for Different Datasets with ADM.

5.3. Simulation Analysis—Class Adjustment: Non-IID Case

We use β to denote the non-IID level. If β = 0, it indicates that data on each MD uniformly belong to labels. Otherwise, if β = 0.8, it means that 80% of the data belongs to one label, and the remaining 20% of the data belongs to other labels. As shown in Table 4, although HDM guarantees accuracy in non-IID settings when each MD has a large dataset, accuracy significantly decreases with smaller datasets. In contrast, ADM considers class distribution at all dataset sizes, reducing larger classes to achieve near-IID conditions and ensuring accuracy. Figure 12a,b demonstrate how ADM adjusts data based on varying data sizes and class distribution across MDs, respectively. MDs 1–10 have 1000 datasets each, while MDs 11–20 have 3000 datasets each. All MDs have non-IID data with β = 0.8. As depicted in Figure 12a, MDs with 1000 datasets reduced to 67% of the existing dataset (represented by the blue bars), while MDs with 3000 datasets decreased even further to 48% of the existing dataset (represented by the green bars). These reductions occurred because ADM adjusts dataset sizes based on the individual sizes of each MD's dataset. Figure 12b illustrates the number of data for each label within the MD with 1000 datasets. Instead of reducing the dataset size for labels with smaller datasets, our approach balances the dataset size with the IID by reducing the data for labels with larger datasets. In conclusion, our proposed ADM effectively adjusts datasets in both IID and non-IID scenarios, alleviating the burden of local training on MDs.

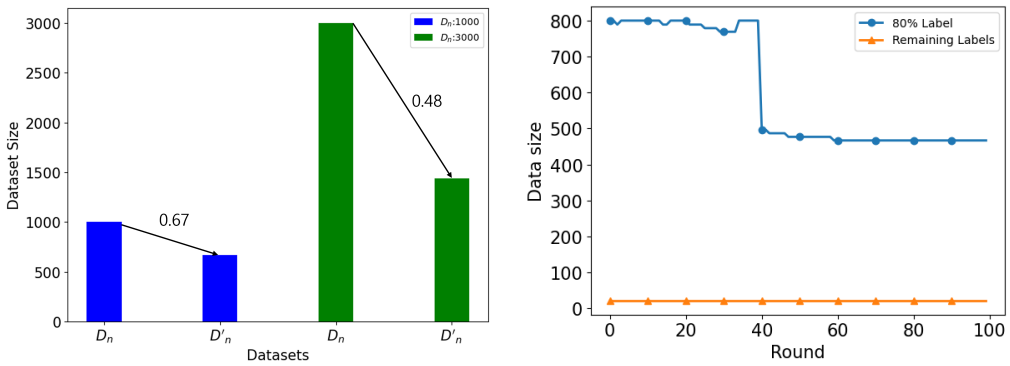


Figure 12. (**a**) Dataset Reduction based on MD's Dataset Size, (**b**) Dataset Reduction based on Class Distribution.

6. Conclusions

In this study, we explored the impact of dataset size on accuracy gain as the communication round evolves. Based on this insight, to alleviate the burden on local training on MDs for FL, we introduced a novel ADM scheme. We incorporated a discount factor to quantify the diminished influence of dataset size on accuracy gain across communication rounds. Specifically, the proposed ADM scheme includes both dataset size adjustment and class adjustment to optimize how dataset reduction is applied across different classes. This optimization takes into account both the discount factor and the KLD. Our experimental results show that our ADM scheme significantly reduces the training burden on MDs while maintaining acceptable training accuracy. In our future work, given that FL remains susceptible to privacy attacks wherein adversaries could potentially retrieve raw data by inspecting local model updates, it is imperative to integrate robust privacy protection mechanisms into our framework (i.e., differential privacy, etc.). Additionally, malicious FL participants pose a significant threat, as they may attempt to inject poisoned or noisy models into an FL server. To mitigate this risk, deploying the proposed scheme on a blockchain platform could be advantageous. Leveraging the inherent integrity of blockchain records ensures that any malicious activities by FL participants can be traced back, as the records remain untampered.

Author Contributions: J.K. and J.B. came up with the ideas and wrote this paper. He developed the suggested ideas and conducted performance evaluations as the first author. Additionally, J.L. supervised the research and assisted with the project as a corresponding author. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2021R1F1A1048098) and in part by the Gachon University research fund of 2022 (GCU- 202300680001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.Y. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečnỳ, J.; Mazzocchi, S.; McMahan, B.; et al. Towards federated learning at scale: System design. *Proc. Mach. Learn. Syst.* **2019**, *1*, 374–388.

3. Lee, J.; Kim, D.; Niyato, D. Market Analysis of Distributed Learning Resource Management for Internet of Things: A Game-Theoretic Approach. *IEEE Internet Things J.* **2020**, *7*, 8430–8439. [CrossRef]
4. Li, H.; Ota, K.; Dong, M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
5. Lee, J.; Kim, D.; Niyato, D. A Novel Joint Dataset and Incentive Management Mechanism for Federated Learning over MEC. *IEEE Access* **2022**, *10*, 30026–30038. [CrossRef]
6. Kim, J.; Kim, D.; Lee, J.; Hwang, J. A Novel Joint Dataset and Computation Management Scheme for Energy-Efficient Federated Learning in Mobile Edge Computing. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 898–902. [CrossRef]
7. Duan, M.; Liu, D.; Chen, X.; Liu, R.; Tan, Y.; Liang, L. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *32*, 59–71. [CrossRef]
8. Zhu, G.; Liu, D.; Du, Y.; You, C.; Zhang, J.; Huang, K. Toward an intelligent edge: Wireless communication meets machine learning. *IEEE Commun. Mag.* **2020**, *58*, 19–25. [CrossRef]
9. Ren, J.; Yu, G.; Ding, G. Accelerating DNN training in wireless federated edge learning systems. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 219–232. [CrossRef]
10. Chen, M.; Yang, Z.; Saad, W.; Yin, C.; Poor, H.V.; Cui, S. A joint learning and communications framework for federated learning over wireless networks. *IEEE Trans. Wirel. Commun.* **2020**, *20*, 269–283. [CrossRef]
11. Amiri, M.M.; Gündüz, D. Federated learning over wireless fading channels. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 3546–3557. [CrossRef]
12. Ko, H.; Lee, J.; Seo, S.; Pack, S.; Leung, V.C. Joint Client Selection and Bandwidth Allocation Algorithm for Federated Learning. *IEEE Trans. Mob. Comput.* **2021**, *22*, 3380–3390. [CrossRef]
13. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. *Proc. Mach. Learn. Syst.* **2020**, *2*, 429–450.
14. Deng, Y.; Lyu, F.; Ren, J.; Zhang, Y.; Zhou, Y.; Zhang, Y.; Yang, Y. SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), Washington, DC, USA, 7–10 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 24–34.
15. Wang, H.; Kaplan, Z.; Niu, D.; Li, B. Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. In Proceedings of the IEEE INFOCOM 2020, Virtual, 6–9 July 2020.
16. Yang, K.; Jiang, T.; Shi, Y.; Ding, Z. Federated Learning via Over-the-Air Computation. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 2022–2035. [CrossRef]
17. Zeng, Q.; Du, Y.; Huang, K.; Leung, K.K. Energy-efficient radio resource allocation for federated edge learning. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
18. Luo, S.; Chen, X.; Wu, Q.; Zhou, Z.; Yu, S. HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6535–6548. [CrossRef]
19. Khan, L.U.; Alsenwi, M.; Yaqoob, I.; Imran, M.; Han, Z.; Hong, C.S. Resource optimized federated learning-enabled cognitive internet of things for smart industries. *IEEE Access* **2020**, *8*, 168854–168864. [CrossRef]
20. Kang, J.; Xiong, Z.; Niyato, D.; Xie, S.; Zhang, J. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet Things J.* **2019**, *6*, 10700–10714. [CrossRef]
21. Ye, D.; Yu, R.; Pan, M.; Han, Z. Federated learning in vehicular edge computing: A selective model aggregation approach. *IEEE Access* **2020**, *8*, 23920–23935. [CrossRef]
22. Lim, W.Y.B.; Xiong, Z.; Miao, C.; Niyato, D.; Yang, Q.; Leung, C.; Poor, H.V. Hierarchical incentive mechanism design for federated machine learning in mobile networks. *IEEE Internet Things J.* **2020**, *7*, 9575–9588. [CrossRef]
23. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated learning with non-iid data. *arXiv* **2018**, arXiv:1806.00582.
24. Yeganeh, Y.; Farshad, A.; Navab, N.; Albarqouni, S. Inverse distance aggregation for federated learning with non-iid data. In *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 150–159.
25. Shirvanian, N.; Shams, M.; Rahmani, A.M. Internet of Things Data Management: A Systematic Literature Review, Vision, and Future Trends. *Int. J. Commun. Syst.* **2022**, *35*, e5267. [CrossRef]
26. Wang, S.; Tuor, T.; Salonidis, T.; Leung, K.K.; Makaya, C.; He, T.; Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1205–1221. [CrossRef]
27. Albaseer, A.; Abdallah, M.; Al-Fuqaha, A.; Erbad, A. Fine-Grained Data Selection for Improved Energy Efficiency of Federated Edge Learning. *IEEE Trans. Netw. Sci. Eng.* **2021**, *9*, 3258–3271. [CrossRef]
28. Jeong, E.; Oh, S.; Kim, H.; Park, J.; Bennis, M.; Kim, S.L. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv* **2018**, arXiv:1811.11479.
29. Chiu, T.C.; Shih, Y.Y.; Pang, A.C.; Wang, C.S.; Weng, W.; Chou, C.T. Semisupervised distributed learning with non-IID data for AIoT service platform. *IEEE Internet Things J.* **2020**, *7*, 9266–9277. [CrossRef]
30. Zhang, W.; Wang, X.; Zhou, P.; Wu, W.; Zhang, X. Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access* **2021**, *9*, 24462–24474. [CrossRef]

31. Zhang, P.; Wang, C.; Jiang, C.; Han, Z. Deep Reinforcement Learning Assisted Federated Learning Algorithm for Data Management of IIoT. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8475–8484. [CrossRef]
32. Lim, W.Y.B.; Luong, N.C.; Hoang, D.T.; Jiao, Y.; Liang, Y.; Yang, Q.; Niyato, D.; Miao, C. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2031–2063. [CrossRef]
33. Grippo, L.; Sciandrone, M. On the Convergence of the Block Nonlinear Gauss–Seidel Method under Convex Constraints. *Operations Research Letters* **2000**, *26*, 127–136. [CrossRef]
34. Beck, A.; Tetruashvili, L. On the Convergence of Block Coordinate Descent Type Methods. *SIAM J. Optim.* **2013**, *23*, 2037–2060. [CrossRef]
35. Kim, J.; Kim, D.; Lee, J. Design and Implementation of Kubernetes Enabled Federated Learning Platform. In Proceedings of the 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 20–22 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 410–412.
36. Shah, A.K.; Oppenheimer, D.M. Heuristics Made Easy: An Effort-Reduction Framework. *Psychol. Bull.* **2008**, *134*, 207. [CrossRef] [PubMed]