Situation-Aware Cluster and Quantization Level Selection Algorithm for Fast Federated Learning

Sangwon Seo, Jaewook Lee, Haneul Ko, *Member, IEEE*, and Sangheon Pack, *Senior Member, IEEE*

***Abstract*—In federated learning (FL), which clients and quantization levels are selected for the deep model parameters has a significant impact on learning time as well as learning accuracy. This is not a trivial issue because it is also significantly affected by factors such as computational power, communication capacity, and data distribution. Considering these factors, we formulate a joint optimization problem for clustering and selecting clusters with quantization levels. Due to the high complexity of the formulated problem, we propose a situation-aware cluster and quantization level selection (SITUA-CQ) algorithm. In this algorithm, the FL server first assembles clients into clusters to mitigate the impact of biased data distributions and determines the most suitable clusters and quantization levels based on their computing power and channel quality. Extensive simulation results show that SITUA-CQ can reduce the round time by up to 80.3% compared to conventional algorithms.**

***Index Terms*—Federated learning (FL), clustering, cluster and quantization level selection**

I. Introduction

With the advances in mobile devices, on-device artificial intelligence (AI) has come to the real world. Federated learning (FL) is perceived as one of the most promising solutions for on-device AI because devices need not share their own data, which indicates that FL can preserve privacy of the device. Moreover, it can reduce communication costs and save the storage on FL server [2]. Due to these benefits, FL has been actively studied in wireless networks [3], [4] and exploited in various fields such as healthcare, voice recognition, object detection applications, and transport infrastructure [2], [5].

The traditional FL procedure [6] is performed as follows. First, the FL server selects the shape of the deep model, the training data type, and various learning parameters. After that, the FL server randomly selects devices as clients, and distributes the deep model with a predetermined quantization level to the selected clients. Note that the conventional FL server allocates a unified quantization level to all selected clients. Next, each client trains the deep model using its own local data and uploads the newly updated deep model to the FL server. Finally, the FL server creates a new deep model by aggregating the deep model from clients. These steps are called round, which is repeated until a target accuracy or a target number of rounds is achieved.

Apparently, each client may have its own training data distribution [7], [8]. If the data distribution differs greatly from the global distribution, a desirable accuracy in the deep model cannot be achieved even after a number of rounds [9], [10]. Therefore, the FL server should select clients whose data distributions are sufficiently close to independent and identically distributed (IID) distributions to achieve the target accuracy [11]. Meanwhile, because most devices may have highly biased data [12], the FL server cannot select a sufficient number of clients with the policy of selecting clients having the data distribution close to IID [13]. Consequently, the learning accuracy can be degraded because the deep model is trained with only a small portion of data. To address this problem, a sequential training scheme has been introduced [14], which clusters are grouped by considering the aggregated data distribution closer to IID and a client in each cluster sequentially train a deep model. However, such sequential training can prolong the overall round time, especially when too many clients are grouped into a single cluster and some of them have a poor performance (i.e., low computing power and/or poor channel conditions) [15], [16]. Consequently, to avoid these undesirable situations, each cluster should be carefully grouped as an appropriate number of clients considering their data distributions and individual round times.

It is also important to assign the most appropriate quantization levels [17] to clusters by considering the estimated cluster round time.[1] For example, if the clients in the clusters have low computing powers and/or poor channel conditions, the cluster round time can be prolonged and thus a low level of quantization should be set to compensate for the increased cluster round time. Meanwhile, if too low quantization levels are employed, sufficiently high accuracy per round cannot be achieved, i.e., more rounds are inevitable to achieve the target accuracy. To summarize, to minimize the total round time while achieving the target accuracy, clients should be properly grouped as clusters. At the same time, appropriate quantization levels should be assigned to the clusters.

To address this problem, we first analyze how cluster and quantization level selections affect the round time and the accuracy per round in different environments. Based on the analysis results, we formulate a joint optimization problem for

S. Seo was with the School of Electrical Engineering, Korea University, Seoul, Korea. S. Seo is currently with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. (e-mail: sw_seo@korea.ac.kr, sangwon@kth.se).

J. Lee is with the Department of Information and Communication Engineering, Pukyong National University, Busan, Korea. (e-mail: jlee0315@pknu.ac.kr).

H. Ko is with the Department of Electronic Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do, 17104, Korea. (e-mail: heko@khu.ac.kr).

S. Pack is with the School of Electrical Engineering, Korea University, Seoul, Korea. (e-mail: shpack@korea.ac.kr, corresponding author: S. Pack).

A preliminary version of this paper has been presented at IEEE Wireless Communications and Networking Conference (WCNC) 2022 [1].

[1]Note that the overall round time is decided by the slowest cluster having the longest cluster round time. In addition, the cluster round time is calculated as the sum of the individual round times of clients in the cluster.

cluster and quantization level selections. To obtain a practical solution to the formulated problem, we develop a situation-aware cluster and quantization level selection (SITUA-CQ) algorithm. In SITUA-CQ, the FL server first estimates the individual round times of clients based on their computing powers and channel conditions. The FL server also groups the clients into several clusters according to their data distributions and estimated round times. After that, the FL server selects the cluster with the smaller cluster round time and allocates appropriate quantization levels, which strikes a balance between the round time and the accuracy per round, to the selected clusters. Simulation results demonstrate that SITUA-CQ can reduce the round time while achieving a desired model accuracy by up to 5.08× faster compared to conventional schemes.

The contributions of this paper can be summarized as follows:

- SITUA-CQ has been motivated by the extensive and realistic evaluations that show the effects of the data distribution, quantization level, and cluster selection on the round time and the accuracy per round.
- Through the joint optimization, SITUA-CQ can significantly reduce the round time while guaranteeing sufficiently high accuracy.
- Extensive simulation results demonstrate that SITUA-CQ provides near-optimal performance.
- SITUA-CQ can be easily implemented in real systems thanks to its low complexity.

The remainder of this paper is organized as follows. Related work is presented in Section II. System model and motivating example are presented in Section III. The optimization problem is formulated in Section IV, then SITUA-CQ is presented in Section V. Simulation results and the concluding remarks are given in Sections VI and VII, respectively.

II. Related Work

Previous studies have been conducted on 1) FL with non-IID distribution [14], [18]–[21]; 2) lightweight deep learning [22]–[27]; and 3) integrated works on both FL and lightweight deep learning [17], [29]–[31].

Several studies [14], [18]–[21] have been conducted to mitigate the accuracy degradation caused by non-IID problem. Wang *et al.* [18] introduced a client selection algorithm based on reinforcement learning in which an FL server selects clients having higher rewards through the improved accuracy. Sattler *et al.* [19] proposed a communication-efficient algorithm that compresses the deep models in uplink and downlink transmissions while reflecting the data distributions of the clients. Duan *et al.* [14] designed a self-balancing algorithm in which the clients augment the data and the FL server schedules the clients to train the deep model with data close to near-IID. Li *et al.* [20] developed a local batch normalization algorithm in which clients exploit the batch normalized layers in the local models with the consideration of the data distributions. In addition, an FL server aggregates the local models except for the batch normalization layers. Zhang *et al.* [21] proposed an adaptive batch size decision algorithm based on reinforcement learning in which the FL server determines the batch size based on the training result in the client.

To speed up the training time of the deep models, some lightweight deep models have been introduced in [22]–[27]. These studies typically employs pruning or quantization techniques [28]. In terms of the pruning technique, Han *et al.* [22] suggested a pruning scheme where the parameters under the threshold of the deep model are removed, and the deep model is retrained to adjust the parameters. Anwar *et al.* [23] proposed another pruning method for the convolutional neural network, which removes less important weight parameters by evaluating their misclassification rates. For quantization, Gupta *et al.* [24] introduced a training scheme on a deep model of 16-bit parameters, which addresses the accuracy degradation problem by adopting stochastic rounding. To further reduce computing time, Wang *et al.* [25] presented another training scheme using 8-bit parameters and implemented a hardware training platform. In addition, Sun *et al.* [26] and Park *et al.* [27] proposed training schemes on the deep model with 4-bit parameters.

To exploit lightweight deep models in FL, several studies [17], [29]–[31] have been conducted in the literature. Abdelmoniem *et al.* [17] proposed a quantization level selection framework in which an FL server randomly selects the clients and determines the quantization level for each client based on a predetermined deadline. Mills *et al.* [29] introduced a method of model quantization in FL where a model is quantized to communicate between an FL server and clients and the deep model is decompressed when training. Mao *et al.* [30] devised a federated learning framework with adaptive pruning in which the FL server determines the pruning rate of each client based on the individual round time. Jhunjhunwala *et al.* [31] suggested an adaptive quantization method to achieve communication efficiency with a low error rate on sending the deep model. The FL server selects the quantization level of the model at each round.

However, there has been no work for an FL system where a lightweight deep learning is adaptively adopted to clients with the consideration of their performance (i.e., computing powers and communication capabilities) and data distribution. Therefore, it takes non-negligible time to obtain sufficiently high accuracy in the FL system.

III. System Model and Motivating Example

In this section, we first describe the system model and present a motivating example through experiments with deep models.

A. System Model

As shown in Figure 1, the system model consists of an FL server and K clients participating in the FL procedure. The FL server and clients can communicate through the base station. The clients have different communication capabilities, computational powers, and data distributions [32].

The round procedure in our FL system consists of six stages: 1) performance reporting, 2) clustering, 3) cluster round time estimation, 4) cluster and quantization level selection, 5) local training, and 6) aggregation. These stages are repeated until a specific number of rounds is reached or a target accuracy is

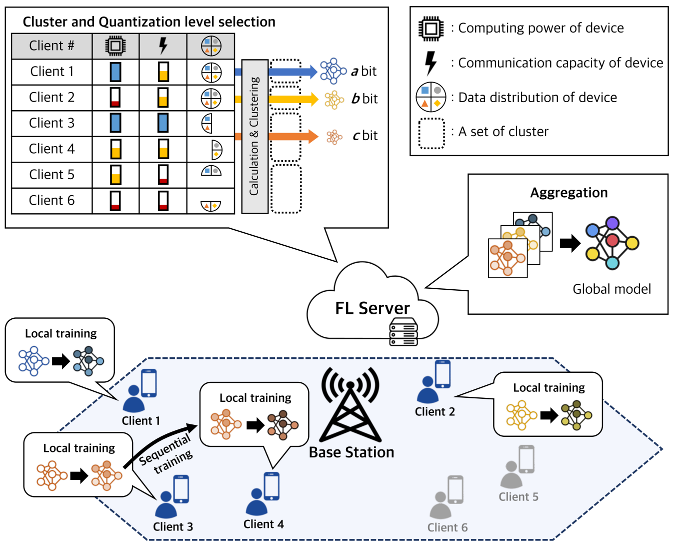

Fig. 1. System model.

achieved. At the performance reporting stage, each client measures its data distribution [14], [33], computational power (in units of floating point operations per second (FLOPS) [34]), and uplink/downlink data rate. After that, each client reports these information to the FL server. Note that these information can be represented with low communication overhead (e.g., only a few bytes).[2] Next, in the clustering stage, the FL server estimates the individual round time of each client. Then, the FL server groups the clients according to their data distributions and the estimated individual round time. Based on the expected cluster round times, the FL server selects the clusters[3] and allocates the quantization levels of the deep models for the chosen clusters (i.e., cluster and quantization level selection stage). After that, the FL server broadcasts the deep model to a client in each cluster. In the local training stage, each client in the selected clusters sequentially trains the deep model using local data. After the local training process is completed in each cluster, the deep model is sent to the FL server. In the final aggregation stage, after the FL server receives all deep models from the selected clusters, it aggregates all deep models for the next round by averaging parameters of deep models.

B. Motivating Example

The accuracy of the deep model is influenced by the data distributions, number of clusters selected, and overall quantization levels (i.e., the distribution of different quantization levels). Meanwhile, the overall quantization levels affect the round time consisting of the upload/download, and the computing time of client. In more detail, the upload/download time is proportional to the quantization level. On the other hand, the computing time is disproportionately affected.

[2]For example, a single probability value can be expressed by two bytes. If the FL system trains the deep model with the dataset having 10 classes (e.g., CIFAR-10), the data distribution can be expressed with only 20 bytes for one client.

[3]In this system, only the selected clusters can participate in the round procedure.

TABLE I. Simulation setting for clients.

Class	Computation		Communication	
	μ	σ	μ	σ
1	100 GFLOPS	5 GFLOPS	66.6 Mbps	2.5 Mbps
2	50 GFLOPS	2.5 GFLOPS	50 Mbps	1.5 Mbps
3	33.3 GFLOPS	1.5 GFLOPS	33.3 Mpbs	1 Mbps
4	25 GFLOPS	1 GFLOPS	17.5 Mbps	0.5 Mbps

TABLE II. Computing time (Unit:ms).

Quantization level	32-bit	16-bit	8-bit
Computing time	38.2	26.6	21

To show the effect of these parameters (i.e., the data distributions, the number of selected clusters, and overall quantization levels) on the round time and the accuracy systematically, we develop an evaluation framework using PyTorch (for computing time) and QPyTorch (for accuracy) [35], which has been released on the Github repository [36]. The detailed evaluation settings are as follows. The deep model consists of six convolutional layers and two fully connected layers. The six convolutional layers have 32, 32, 64, 64, 128, and 128 filters, respectively, and ReLu and the max-pooling layers which filter size is 2×2 are followed. Then, the fully connected layers with 512 units and 10 units (i.e., final output layer) are followed. The CIFAR-10 dataset is exploited for obtaining the accuracy result [37]. In CIFAR-10, there are 10 classes, and it consists of 50,000 training images and 10,000 test images.

The data distributions of clients are generated by the Dirichlet distribution, $Dir(v)$, [38] where v is a vector of parameters and the length of v is the same as the number of classes in a dataset. To measure the distribution distance, we use the Kullback-Leibler divergence, D_{KL}, which represents the distance between the cluster distribution and the target distribution (i.e., a uniform distribution). The process of setting the data for the clients is as follows. First, we set the range of the distribution distance (i.e., the minimum and maximum values of D_{KL}) and v. After that, the distribution generation process is conducted until the number of distributions, which distances are in the range of the minimum and maximum value, is equal to the number of clients. In our experiment, we use the distributions that satisfy the overall distribution distance lower than 0.05 (i.e., near-IID of overall distribution distance). Our implementation for generating the distributions can be founded at our GitHub repository [36]. Meanwhile, the simulation setting for clients is summarized in Table I. We assume that the clients are classified into four classes with diverse computation and communication capabilities [32]. Specifically, the computation and communication capabilities of each class are assumed to follow Gaussian distributions with μ and σ in Table I. Meanwhile, the number of clients, K, is set to 100 and the default number of clients of one cluster is two.

Table II shows computing time according to quantization levels. To measure the computing time, a computer with Intel i7-8700 CPU, 16GB RAM, and NVIDIA GeForce 1060Ti is used. From Table II, it can be found that the deep models with 16-and 8-bit parameters can reduce the computing time by 30% and 45%, respectively. That is, if the FL server allocates lower bit parameters to specific clients, the client can significantly reduce its computing time.

Figure 2 shows the effect of the overall data distributions of clusters on the accuracy. When generating the distributions for clients, the range of D_{KL} is set to 0.01. For example, if D_{KL} is 0.2, the minimum and maximum values of D_{KL} are respectively set to 0.195 and 0.205 (i.e., the distribution distance of each client is in the range from 0.195 to 0.205). In this result, 40% of the clusters are selected in each round, and 8- and 32-bit parameters are evenly employed. It can be found that the accuracy increases rapidly with the decrease of D_{KL}. However, when D_{KL} is smaller than 0.4, no significant improvement is observed in terms of the accuracy. In addition, if D_{KL} becomes close to 0 (i.e., IID), the fluctuation of the accuracy is mitigated. This result indicates that the FL server should group the clients into clusters to attain sufficiently short distribution distance (i.e., D_{KL}) in each cluster.

Figure 3 shows the effect of the number of selected clusters on the accuracy. In this result, the FL server selects the clusters with faster cluster round time in each round. Then, 8- and 32-bit parameters are evenly allocated to the selected clusters and the average D_{KL} of the clusters is set to 0.4. When selecting a small number of clusters (e.g., 2, 4, and 5 clusters), since the FL server selects the clusters with faster cluster round time, the overall FL procedure can be finished in a short time. However, the accuracy cannot converge to a sufficiently high level. Moreover, the overfitting problem (i.e., the accuracy decreases according to the round) can occur if 2 clusters are selected. Except for the case, the accuracy increases when more and more clusters are selected, but is almost saturated if the number of selected clusters is over 20 (i.e., 40% of total clients). Thus, a sufficient number of clusters (e.g., over 20 clusters in this result) should be selected by considering the effect on the accuracy and the round time.

Figure 4 shows the effect of the distribution of quantization levels on the accuracy. Each cluster is composed of 5 clients and thus 20 clusters participate in the FL procedure. The FL server selects 8 clusters at each round and the average D_{KL} of the clusters is set to 0 (i.e., IID distribution). Before 10,000s, no significant discrepancy is observed depending on the distribution of quantization levels. However, after 10,000s, it can be found that the accuracy is gradually increased if a high proportion of quantization levels (i.e., 32-bit parameters) is included. For example, when all quantization levels are 8-bit parameters, the accuracy is bounded to 81.3%. On the other hand, if only 32-bit parameters are included, the upper bound of accuracy can be increased to 84.4%. Consequently, the FL server should determine appropriate distribution of quantization levels according to the effect on the accuracy and the round time.

To summarize, to reduce the round time while guaranteeing the target accuracy, sufficiently short distribution distances of each cluster should be guaranteed. In addition, a sufficient number of clusters should be selected and suitable distributions of quantization levels should be determined.

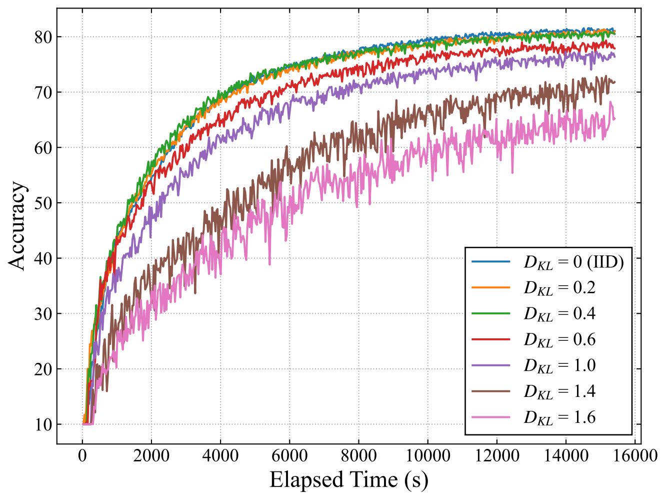

Fig. 2. Effect of the overall data distributions of clusters.

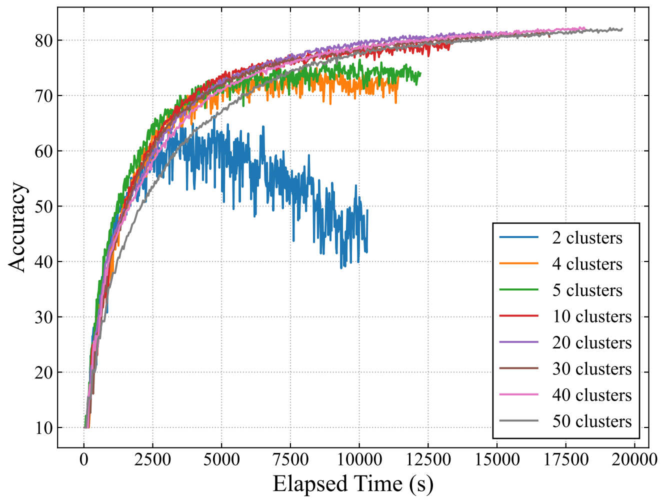

Fig. 3. Effect of the number of selected clusters.

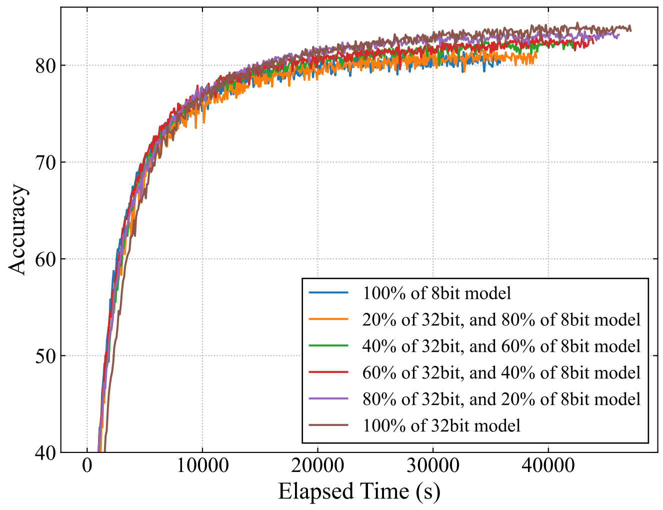

Fig. 4. Effect of the distribution of quantization levels.

IV. Problem Formulation

In this work, the optimization problem is formulated in terms of grouping the clients into clusters, selecting clusters, and allocating quantization levels. To minimize the round time while guaranteeing the target accuracy, the clients are grouped into clusters according to their performances (i.e., computing powers and communication capabilities) and data distributions. In addition, the FL server selects several clusters and allocates quantization levels for the selected clusters by considering the effects on the accuracy and round time. Key notations are listed in Table III.

First of all, let $\mathbf{A}$ and A_i be the set of all possible combinations from the clusters and the ith combination in $\mathbf{A}$, respectively. Then, $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \{A_1, A_2, A_3, ..., A_i, ..., A_I\} \tag{1}$$

where I is the total number of combinations, i.e., the set of selected clusters. I is calculated as $I = 2^K - 1$. For example, if there are three clusters (i.e., Cluster 1, Cluster 2, and Cluster 3), 7 combinations of the clusters can be defined and thus we have $\mathbf{A}$ = {{Cluster 1}, {Cluster 2}, {Cluster 3}, {Cluster 1, Cluster 2}, {Cluster 1, Cluster 3}, {Cluster 2, Cluster 3}, {Cluster 1, Cluster 2, Cluster 3}}). In this case, for instance, A_4 is given {Cluster 1, Cluster 2}.

When conducting the FL procedure, the individual round time consists of the download, training, and upload times. That is, the individual round time, $R^C_{k,b,t}$, can be calculated as

$$R^C_{k,b,t} = R^D_{k,b,t} + R^P_{k,b,t} + R^U_{k,b,t} \tag{2}$$

where $R^D_{k,b,t}$ and $R^U_{k,b,t}$ are the download and upload times of client k learning the deep model with b-bit parameters in round t, respectively. In addition, $R^T_{k,b,t}$ denotes the computing time of client k learning the deep model with b-bit parameters in round t.

Since the download time is influenced by the size of the deep model W_b and downlink data rate $d_{k,t}$, $R^D_{k,b,t}$ can be computed as

$$R^D_{k,b,t} = \frac{W_b}{d_{k,t}}. \tag{3}$$

Similarly, the upload time can be obtained by dividing W_b by the uplink data rate, which is given by

$$R^U_{k,b,t} = \frac{W_b}{u_{k,t}}. \tag{4}$$

Meanwhile, the computing time is determined by the required number of computations of the deep model (denoted by F) and FLOPS (i.e., the number of available computations per second) of client k in round t (denoted by $p_{k,t}$). Moreover, because we consider the quantization levels in the system model, the computing time is influenced in non-linear according to the allocated quantization level. Therefore, the computing time should be adjusted with the normalized computing time according to b-bit parameter. Thus, $R^P_{k,b,t}$ is given by

$$R^P_{k,b,t} = \alpha(F, p_{k,t}, b) \tag{5}$$

TABLE III. Summary of notations.

Notation	Description
A_i	Combination of clusters
b	Bit index
i	Combination index
j	Cluster index
k	Client index
t	Round index
$x_{i,t}$	Binary decision variable on whether the combination A_i is selected in round t
$y_{j,k,t}$	Binary decision variable on whether client k is grouped into cluster j in round t
$z_{j,b,t}$	Binary decision variable on whether cluster j is allocated the deep model with b-bit parameters in round t
W_b	Size of the deep model with b-bit parameters
$d_{k,t}$	Downlink data rate of client k in round t
$p_{n,t}$	computational power (FLOPS) of client k in round t
$u_{k,t}$	Uplink data rate of client k in round t
F	Required number of computations for training the deep model
$n_{b,k}$	Normalized computing time of client k with b-bit parameters
θ_k	Threshold for the number of clients in selected clusters
θ_d	Threshold for distribution distance
θ_b	Threshold for the proportion of b-bit parameters

where $\alpha(F, p_{k,t}, b)$ is the function to calculate the computing time of client k with b-bit parameters in round t. $\alpha(F, p_{k,t}, b)$ can be defined as

$$\alpha(F, p_{k,t}, b) = \frac{F}{p_{k,t}} \times n_{b,k} \tag{6}$$

where $n_{b,k}$ is the normalized computing time of b-bit parameters in client k.

Note that a deep model is sequentially trained by clients in each cluster to avoid any accuracy degradation caused by skewed data distributions. Hence, the cluster round time $R_{j,b,t}$ is given by the summation of the individual round times of all clients in cluster j, which can be represented as

$$R_{j,b,t} = \sum_{b}^{B} \sum_{k}^{K} (R^C_{k,b,t} \times y_{j,k,t} \times z_{j,b,t}) \tag{7}$$

where $y_{j,k,t}$ denotes a binary decision variable on whether client k belongs to cluster j in round t. $z_{j,b,t}$ is a binary decision variable on whether cluster j is allocated the deep model with b-bit parameters in round t. B and K are the biggest bit size among the available bit parameters and the number of clients, respectively.

The objective of this paper is to minimize the round time while maintaining sufficiently high accuracy. For this objective, we make three types of decisions: 1) decision on how to group the clients into clusters; 2) decision on which combination of clusters are selected; and 3) decision on the quantization levels for the selected clusters. The first and third decisions can be handled with the binary decision variables, $y_{j,k,t}$ and $z_{j,b,t}$, respectively. Meanwhile, for the second type of decision, we define $x_{i,t}$ as another binary variable on whether the ith combination, A_i, is selected at round t. Then, the objective function and constraints can be described as follows:

$$\min_{x,y,z}\left(\sum_{t}^{T}\sum_{i}^{I}x_{i,t}\times\max_{j\in A_i}\sum_{b}^{B}\sum_{k}^{K}(R^{C}_{k,b,t}\times y_{j,k,t}\times z_{j,b,t})\right) \tag{8}$$

$$\text{s.t.}\ \sum_{i}^{I}x_{i,t}=1,\forall t, \tag{8a}$$

$$\sum_{j}^{J}y_{j,k,t}=1,\forall k,\forall t, \tag{8b}$$

$$\begin{cases}\sum_{b}^{B}z_{j,b,t}=1, & \text{if } j\in A_i \text{ and } x_{i,t}=1\\ \sum_{b}^{B}z_{j,b,t}=0, & \text{otherwise}\end{cases}, \tag{8c}$$

$$D_{KL}(P_j||P_T)\le\theta_d,\forall j\in A_i \text{ and } x_{i,t}=1, \tag{8d}$$

$$\zeta_{A_i,t}\ge\theta_k,\forall A_i, \tag{8e}$$

$$\psi_{A_i,b,t}\ge\theta_b,\ \forall b\in B,\forall A_i. \tag{8f}$$

The FL server can select a single combination A_i at each round t with the constraint (8a). The constraint (8b) can be satisfied if a client needs to belong to one cluster. The FL server can allocate one of the deep models of b-bit parameters to one cluster with the constraint (8c). As discussed in Section III-B, the data distribution of each cluster significantly affects the accuracy of the deep model. To guarantee sufficiently high accuracy, the distance between the cluster distribution (i.e., P_j) and the target distribution (i.e., P_T) should be equal to or smaller than a threshold for distribution distance, θ_d. The corresponding constraint can be expressed with the Kullback-Leibler divergence $D_{KL}(P_j||P_T)$ representing the distance between the cluster distribution (i.e., P_j) and the target distribution (i.e., P_T) as in (8d). Meanwhile, to avoid any overfitting issues, diverse and a sufficient number of clients in the selected clusters should be chosen by constraint (8e), where θ_k is a threshold for the number of clients in the selected clusters. $\zeta_{A_i,t}$ is the number of clients in combination A_i, and it is given by

$$\zeta_{A_i,t}=\sum_{j\in A_i}\sum_{k}^{K}\sum_{b}^{B}y_{j,k,t}\times z_{j,b,t}. \tag{9}$$

Meanwhile, as revealed in Figure 4, sufficiently high accuracy cannot be achieved if the FL server allocates lower bit parameters (e.g., 8-bit parameters) aggressively. Therefore, the proportion of b-bit parameters in the combination A_i at round t, $\psi_{A_i,b,t}$, should be equal to or larger than a threshold θ_b for the proportion of b-bit parameters, which is given in (8f). Meanwhile, $\psi_{A_i,b,t}$ can be obtained as

$$\psi_{A_i,b,t}=\frac{\sum_{j\in A_i}\delta[z_{j,b,t}=1]}{|A_i|} \tag{10}$$

where $\delta[\cdot]$ is a delta function that returns 1 if the given condition is true. Otherwise, it gives 0. The formulated optimization problem can be resolved by a brute-force search. Since the numbers of binary decision variables (i.e., $x_{i,t}$, $y_{j,k,t}$, and $z_{j,b,t}$) are I, $2K$, and $K+B$, the complexity of the formulated problem is given by $\mathcal{O}(2^{3K+I+B})$.

V. SITUA-CQ

Algorithm 1 SITUA-CQ

```
1: Initialize: the sets $S_j$, $U_C$, $A_S$, $U_P$, and $U_B$
2: Receive performance information from clients
3: $j \leftarrow 0$, $k \leftarrow 0$
4: $z_{j,b,t} \leftarrow 0,\ \forall j,b,t$
5: while $U_P \neq \emptyset$ do
6:     $k^* \leftarrow \arg\min_{k\in U_P} D_{KL}((P_{S_j}+P_k)||P_T)$
7:     Add $k^*$ to $S_j$
8:     Delete $k^*$ from $U_P$
9:     if $D_{KL}(P_{S_j}||P_T) \le \theta_d$ then
10:        Add set $S_j$ to $U_C$
11:        $j \leftarrow j+1$
12:    end if
13: end while
14: $l \leftarrow 0$
15: while $l \le \theta_k$ do
16:     $j^* \leftarrow \arg\min_{S_j\in U_C} R_{j,B,t}$
17:     $y_{j^*,k,t}=1,\ \forall k\in S_{j^*}$
18:     Add $S_{j^*}$ to $A_S$
19:     Remove $S_{j^*}$ from $U_C$
20:     $l \leftarrow l+|S_{j^*}|$
21: end while
22: $m \leftarrow 0$, $b^* \leftarrow \arg\max_{b\in U_B} b$
23: Remove $b^*$ from $U_B$
24: while $m \le |A_S|$ do
25:     $j^* \leftarrow \arg\min_{S_j\in A_S} R_{j,b^*,t}$, s.t. $\sum_{b\in U_B} z_{j^*,b,t}=0$
26:     $z_{m,b^*,t}=1$
27:     if $\sum_{n=b^*}^{B}\psi_{A_S,n,t} \ge \sum_{n=b^*}^{B}\theta_n$ then
28:        $b^* \leftarrow \arg\max_{b\in U_B} b$
29:        Remove $b^*$ from $U_B$
30:     end if
31:     $m \leftarrow m+1$
32: end while
```

Since the complexity of the formulated optimization problem is too high, it is not feasible to use in the FL server. Therefore, we propose the SITUA-CQ algorithm to find a near-optimal solution by jointly considering the accuracy and round time. Algorithm 1 shows the detailed procedure of SITUA-CQ which consists of an initialization part and three loops. In the first loop, the algorithm clusters the clients according to their data distributions, which maintains the distribution distance in each cluster to be close to IID. Next, in the second loop, the algorithm selects clusters having faster cluster round times until the number of clients is satisfied, which is to attain sufficiently high accuracy while the round time is not much prolonged. In the third loop, the algorithm allocates quantization levels to the selected clusters considering the predetermined distributions of quantization levels, which speeds up for clusters having long round times while maintaining high accuracy.

At the initialization part, the set of clustered clients S_j, the set for whole clusters U_C, and the set for selected clusters A_S are initialized as empty sets. The set of whole clients U_P, and the set of available parameters in deep model U_B are loaded (line 1 in Algorithm 1). Then, the FL server receives some information from clients including the number of operations per second, uplink/downlink data rate, and the data distribution (line 2 in Algorithm 1).

In the first loop, a client k^* in U_P is selected, which has the lowest $D_{KL}((P_{S_j} + P_k)||P_T)$ to guarantee sufficiently high accuracy (line 6 in Algorithm 1). Then, k^* is added to S_j and deleted from U_P (lines 7-8 in Algorithm 1). If the distance between the distribution of S_j and the target distribution (i.e., $D_{KL}(P_{S_j}||P_T)$) is lower than θ_d, S_j is added to U_C (lines 9-10 in Algorithm 1) and j is incremented to check another cluster in the next iteration (line 11 in Algorithm 1).

In the second loop, Algorithm 1 selects clusters until the number of selected clients l is equal to or less than θ_k (line 15 in Algorithm 1). First, a cluster j^* with the shortest cluster round time $R_{j,B,t}$ is chosen from U_C (line 16 in Algorithm 1). Note that it is assumed when the biggest parameter (i.e., B) is allocated to mitigate the calculation overhead. The selected cluster j^* is added to A_S and deleted from from U_C (lines 18-19 in Algorithm 1). After that, the algorithm selects the highest b-bit parameter to allocate the highest one to the faster clusters (line 22 in Algorithm 1).

In the third loop, quantization levels are selected for all selected clusters (line 24 in Algorithm 1). A cluster j^* with the shortest cluster round time is selected from A_S (line 25 in Algorithm 1). Next, the deep model of b^*-bit parameters is allocated to the selected cluster j^* (line 26 in Algorithm 1). After that, the ratio of quantization levels is checked (line 27 in Algorithm 1). If the ratio is guaranteed (i.e., $\sum_{n=b^*}^{B} \psi_{A_S,n,t} \geq \sum_{n=b^*}^{B} \theta_n$), b^* is changed to the next highest parameter to avoid any increase in round time (lines 28-30 in Algorithm 1).

Algorithm 1 consists of three loops and its complexity is decided by a loop with the highest complexity. The first loop repeats K^2 operations to group the clients into proper clusters. In addition, the second loop repeats K operations to select appropriate clusters. The third loop also repeats K operations to allocate the quantization levels for the selected clusters. To sum up, the first loop has the highest complexity, and thus the complexity of our algorithm is given by $\mathcal{O}(K^2)$, which is significantly lower than that for solving the formulated optimization problem using the brute-force approach (i.e., $\mathcal{O}(2^{3K+I+B})$).

In order to implement the proposed algorithm, some information (i.e., data distribution, FLOPS, and uplink/downlink data rate of the clients) should be collected at the FL server. This information can be easily collected by the clients and delivered to the FL server with a low communication overhead. Therefore, the proposed algorithm can be comfortably implemented in real systems. Specifically, each client can obtain its data distribution by counting the numbers of data in each class. In addition, each client has inherent FLOPS information that can be tactically measured by using a simple library (e.g., the flops profiler function of DeepSpeed package [39]). Meanwhile, uplink/downlink data rate can be calculated based on the CQI reporting in cellular systems. All collected information can be represented with only a few bytes (e.g., 20 bytes for data distribution) and therefore only negligible communication overhead is required.

VI. Simulation Results

For the performance evaluation, we have developed a PyTorch and QPyTorch-based simulator and conducted more than 100,000 simulations to obtain reliable results. SITUA-CQ is compared with the following schemes.

1) RANDOM [6]: The FL server groups each client into one cluster. Then, it randomly selects θ_k clusters and allocates 32-bit parameters for the chosen clusters.
2) RANDQL [17]: The FL server groups each client into one cluster. Then, it randomly selects θ_k clusters and determines quantization levels for the chosen clusters based on their cluster round time.
3) DISTA: The FL server groups each client into one cluster. In addition, the FL server selects θ_k clusters having shorter cluster round times while providing the distribution distances lower than the threshold θ_d. Then, the FL server allocates a deep model with 32-bit parameters to them.
4) CLUST [14]: The FL server groups five clients into one cluster to minimize the distribution distance in each cluster. The FL server randomly selects a fixed number of clusters (i.e., 60% among total clusters) and allocates a deep model with 32-bit parameters.

The default setting for the experiments is as follows. The number of clients, K, is set to 100 and the lower bounds of the number of selected clients, θ_k, is set 60 [40]. The computation power, $p_{k,t}$, and uplink/downlink data rate, $u_{k,t}$ and $d_{k,t}$, of clients are set according to Table I in Section III. To assess the accuracy of SITUA-CQ, we exploit the CIFAR-10 dataset. In addition, deep models of 32, 16, and 8-bit parameters are exploited. The shapes of the deep models are described in Section III and the required computational power of the deep model F is 400 GFLOPS, which is measured by the DeepSpeed package [39]. The normalized computing times $n_{b,k}$ of 32, 16, and 8-bit parameters are set to 1, 0.7, and 0.55 for all k, respectively [41]. The size of the deep models, W_b, of 32, 16, and 8-bit parameters are respectively 20, 10, and 5 MB. The distribution of quantization levels, θ_b, of 32, 16, 8-bit parameters are set to 0.3, 0.3, and 0.4, respectively.

A. Optimal vs. SITUA-CQ

Table IV shows the average computation time for finding the solution in a single round. This result is obtained using an Intel i9-10900X CPU and 64GB RAM. From Table IV, it can be found that the computation time of the optimal solution increases drastically as K increases. Even when K is small (e.g., 5 or 6), the optimal solution consumes significant computation time and we could not measure the computation

TABLE IV. Computation time (Unit: ms): Optimal vs. SITUA-CQ.

K	5	6	10	50	100
Optimal	3855.95	192507.77	N/A	N/A	N/A
SITUA-CQ	0.74	0.97	2.25	35.56	125.62

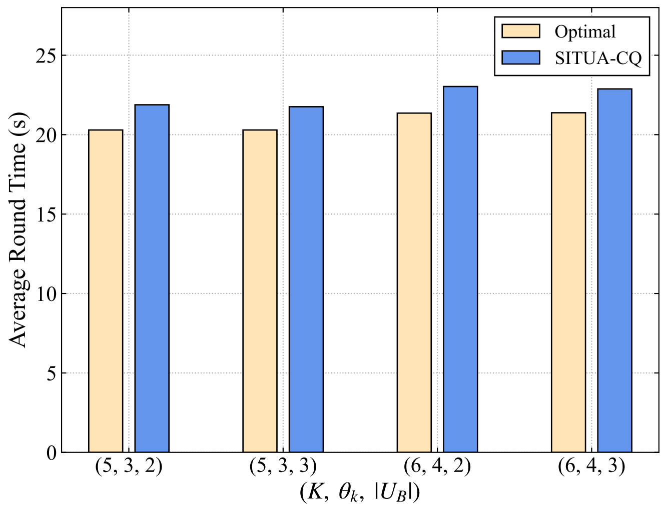

Fig. 5. Average round time of Optimal and SITUA-CQ.

time when K exceeds 6. In contrast, SITUA-CQ maintains a low computation time of less than 1ms when K does not exceed 6. Although K is larger (e.g., 100), SITUA-CQ can guarantee a short computation time of 125.62ms, which is only about 0.5% of the average round time.

Figure 5 shows the average round time under different combinations of the number of clients, the minimum number of selected clients, and the number of available quantization levels (i.e., K, θ_k, and $|U_B|$). Note that only $K = 5$ and $K = 6$ are evaluated due to high complexity of the optimal problem. Meanwhile, θ_k is set to 3 or 4, and $|U_B|$ is set to 2 (i.e., 32- and 16-bit parameters) or 3 (i.e., 32-, 16-, and 8-bit parameters). From Figure 5, it can be observed that the SITUA-CQ can provide a round time comparable to that of the optimal solution, regardless of K and $|U_B|$. Specifically, the difference between SITUA-CQ and the optimal solution is $7.0\% \sim 7.8\%$.

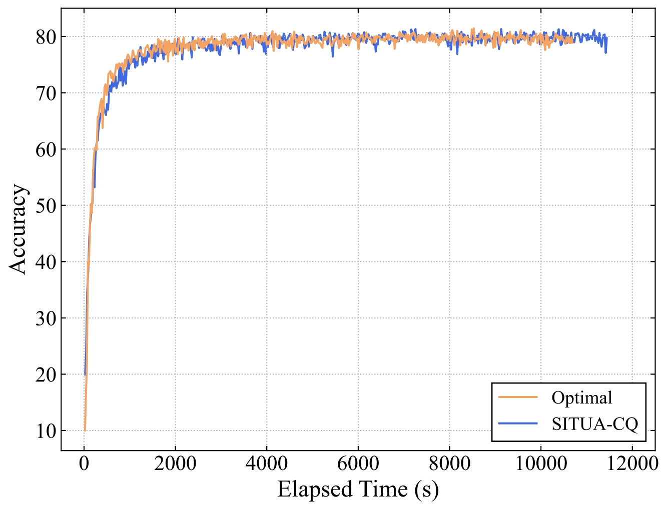

Fig. 6. Accuracy curves of Optimal and SITUA-CQ.

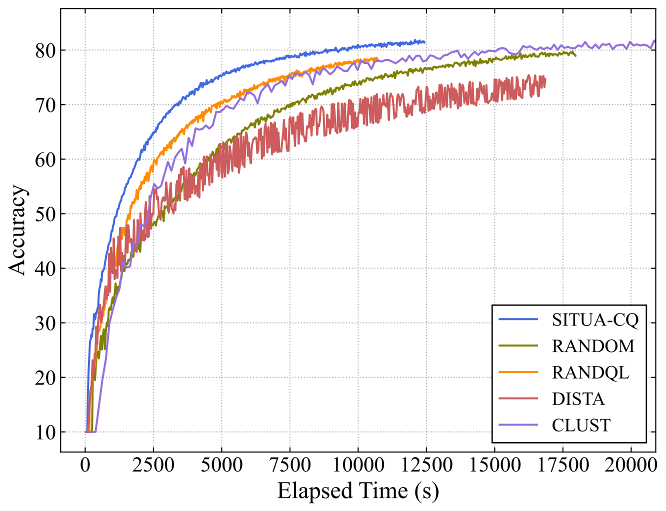

Fig. 7. Accuracy curves of SITUA-CQ and comparison schemes.

Figure 6 compares the accuracy of the optimal solution and SITUA-CQ when K and $|U_B|$ are set to 6 and 3, respectively. Note that the accuracy of the optimal solution and SITUA-CQ is measured during the same rounds (i.e., 500 rounds). From Figure 6, it can be found that the accuracy of the optimal solution and SITUA-CQ is saturated after approximately 4,000s. Furthermore, SITUA-CQ and the optimal solution have the values of the highest accuracy at 81.33% and 81.44%, respectively. To sum up, SITUA-CQ can achieve comparable saturation time and accuracy to the optimal solution.

B. *Effect of Accuracy*

Figure 7 shows the accuracy of SITUA-CQ and the comparison schemes.As shown in Figure 7, SITUA-CQ can achieve the highest accuracy (i.e., 81.8%) while finishing 500 rounds at 12,425s. This can be explained as follows. First, SITUA-CQ clusters the clients to maintain the distribution distance in each cluster to be close to IID. Then, SITUA-CQ selects an appropriate number of clusters to avoid overfitting problems and prolonged round time.[4] In addition, SITUA-CQ allocates quantization levels to each selected cluster according to the cluster round time. For example, if the round time of a specific cluster is relatively longer than that of other clusters, low quantization level is allocated to that cluster.

Meanwhile, even though CLUST can achieve comparable accuracy (i.e., 81.5%) to SITUA-CQ, this accuracy can be achieved after long elapsed time (i.e., 19,957s). This is because CLUST selects too many clusters in each round, and the FL server waits the updated deep models from all clients in

[4]Note that, if too small number of clients are included in the selected clusters, the overfitting problem can occur. On the other hand, if too large number of clients are in the selected clusters, the round time can increase.

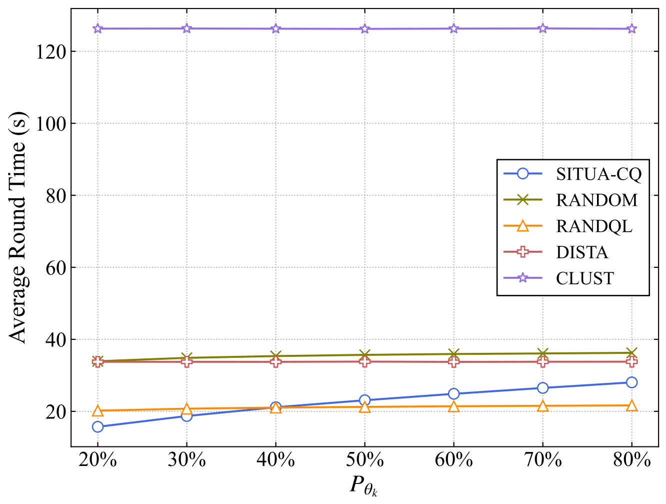

Fig. 8. Effect of the number of selected clients.

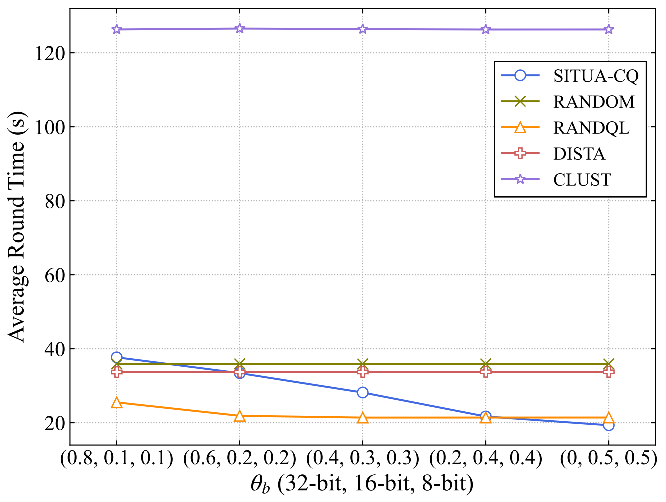

Fig. 9. Effect of the distributions of quantization levels.

those clusters. RANDOM and RANDQL show respectively low accuracy of 79.7% and 78.6% since they do not consider the distribution distance. Interestingly, DISTA has the lowest accuracy which also highly fluctuates considerably. This is because only a few clients participate in the FL procedure of DISTA.

C. *Effect of P_{θ_k}*

Figure 8 shows the effect of the proportion between the minimum number of selected clients (denoted by θ_k) and the number of clients (denoted by K), P_{θ_k}, on average round time. As shown in Figure 8, the average round time of SITUA-CQ increases slightly with an increment of P_{θ_k} whereas those of the other schemes are rarely influenced by P_{θ_k}. In more detail, the average round time of SITUA-CQ is the lowest among the schemes when $P_{\theta_k} \leq 40\%$. On the other hand, if P_{θ_k} exceeds 40%, RANDQL shows a shorter round time than SITUA-CQ. This is because RANDQL does not conduct any (time consuming) sequential training among clients. However, note that the average round time of SITUA-CQ does not exceed those of RANDOM, DISTA, and CLUST regardless of P_{θ_k}, despite of sequential training in clusters. This is because SITUA-CQ first composes each cluster until it satisfies the data distance of the cluster. At this time, the FL server constructs the next cluster even if the cluster satisfies data distance with a small number of clients (e.g., one or two clients). After the FL server finishes clustering, it selects the clusters having faster cluster round time and allocates quantization levels to each cluster according to cluster round time.

D. *Effect of θ_b*

Figure 9 shows the effect of the distribution of quantization levels, θ_b, on the average round time. From Figure 9, more 32-bit parameters are involved (e.g., θ_b of 32-bit parameters is 0.8), the average round time of SITUA-CQ is larger than those of the comparison schemes except CLUST. This is because RANDOM, RANDQL, and DISTA do not conduct any sequential training. Moreover, if θ_b is high, SITUA-CQ cannot avoid a situation where 32-bit parameters are allocated to the selected clusters having poor performance clients. On the contrary, less 32-bit parameters are used, the average round time of SITUA-CQ decreases drastically.

It can be also seen that the improvement in SITUA-CQ is saturated when θ_b of 32-bit parameters exceeds a specific value (e.g., 0.2 in Figure 9). Meanwhile, RANDQL is saturated when θ_b of 32-bit parameters exceeds a value (e.g., 0.6 in Figure 9). In other words, RANDQL is more quickly saturated than SITUA-CQ because all of the selected clients in RANDQL train the deep model in a parallel manner. Consequently, RANDQL has a smaller difference in $R_{j,b,t}$ than SITUA-CQ.

E. *Effect of Client Classes*

Figure 10 shows the effect of different client classes where the average round time of SITUA-CQ is normalized as one. Depending on the case, different combinations of client classes (in Table I) are considered and the number of clients in each case is summarized in Table V. For each case, RANDQL and CLUST show the lowest and highest normalized round times, respectively. Interestingly, the performance gap between RANDQL and SITUA-CQ for Case 4 is smaller than those for Cases 2 and 3. In addition, RANDOM, DISTA, and CLUST exhibit the apparently increased normalized round time for case 4. To summarize, it can be concluded that the performance improvement of SITUA-CQ is significant when heterogeneous client classes are considered. That is,

TABLE V. The number of clients in each class according to the case.

Client Class	Case 1	Case 2	Case 3	Case 4
1	100	50	33	25
2	0	50	33	25
3	0	0	34	25
4	0	0	0	25

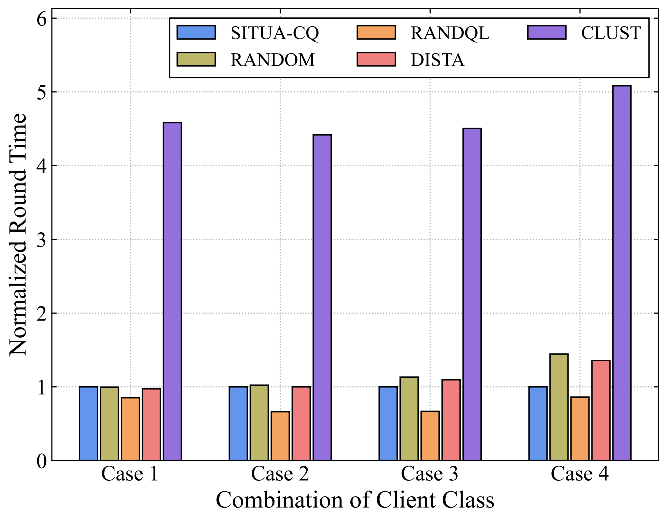

Fig. 10. Effect of the client classes.

quantization level selection in SITUA-CQ reduces the round time of clusters having slower round times. At this time, when cluster round times among selected clusters do not vary significantly, the effect of quantization levels on the round time is not remarkable. However, since the cluster round times among selected clusters are quite different in a converse situation (i.e., client classes are varied), the effect of quantization level selection on the round time can be improved.

VII. Conclusion

In this paper, we proposed the SITUA-CQ algorithm, which jointly addresses clustering, cluster selection, and quantization level selection problems with the consideration of distribution distance, computing powers, and communication conditions. SITUA-CQ shows not only comparable performance compared to the optimal solution but also significantly short computation time. Simulation results demonstrate that SITUA-CQ can guarantee sufficiently high accuracy with a competitive round time for the comparison schemes. Moreover, SITUA-CQ shows a remarkable improvement when heterogeneous clients are considered. In future work, we will devise a multiple learning task-based cluster and quantization level selection and scheduling algorithm to minimize the round times of multiple tasks simultaneously.

Acknowledgement

This research was supported by National Research Foundation (NRF) of Korea Grant funded by the Korean Government (MSIT) (No. 2021R1A4A3022102 and 2020R1A2C3006786).

References

[1] S. Seo, J. Lee, H. Ko, and S. Pack, “Performance-Aware Client and Quantization Level Selection Algorithm for Fast Federated Learning,” in *Proc. IEEE WCNC 2022*, April 2022.

[2] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, “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.

[3] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy Efficient Federated Learning Over Wireless Communication Networks,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935-1949, March 2021.

[4] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, ”A Joint Learning and Communications Framework for Federated Learning Over Wireless Networks,” in IEEE Transactions on Wireless Communications, vol. 20, no. 1, pp. 269-283, Jan. 2021.

[5] 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.

[6] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *Proc. AISTATS 2017*, April 2017.

[7] D. Verma, G. White, S. Julier, S. Pasteris, S. Chakraborty, and G. Cirincione, “Approaches to Address The Data Skew Problem in Federated Learning,” in *Proc. Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, May 2019.

[8] S. A. Tijani, X. Ma, R. Zhang, F. Jiang and R. Doss, ”Federated Learning with Extreme Label Skew: A Data Extension Approach,” 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–8, doi: 10.1109/IJCNN52387.2021.9533879.

[9] H. Ko, J. Lee, S. Seo, S. Pack, and V. Leung, “Joint Client Selection and Bandwidth Allocation Algorithm for Federated Learning,” *IEEE Transactions on Mobile Computing*, to appear.

[10] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, “The Non-IID Data Quagmire of Decentralized Machine Learning,” arXiv preprint arXiv:1910.00189, October 2019.

[11] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated Learning with Non-IID Data,” arXiv preprint arXiv:1806.00582v1, June 2018.

[12] V. Kulkarni, M. Kulkarni, and A. Pant, “Survey of Personalization Techniques for Federated Learning,” in *Proc. WorldS4 2020*, July 2020.

[13] T. Huang, W. Lin, 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.

[14] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, “Self-Balancing Federated Learning With Global Imbalanced Data in Mobile Systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 1 Jan. 2021.

[15] T. Nishio and R. Yonetani, “Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge,” in *Proc. IEEE ICC 2019*, May 2019.

[16] J. Lee, H. Ko, and S. Pack, “Adaptive Deadline Determination for Client Selection in Federated Learning,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 3367–3371, March 2022.

[17] A. M. Abdelmoniem, and Marco Canini, “Towards Mitigating Device Heterogeneity in Federated Learning via Adaptive Model Quantization,” in *Proc. EuroMLSys 2021*, April 2021.

[18] H. Wang, Z. Kaplan, D. Niu, and B. Li, “Optimizing Federated Learning on Non-IID Data with Reinforcement Learning,” in *Proc. IEEE INFOCOM 2020*, July 2020.

[19] F. Sattler, S. Wiedemann, K. -R. Müller, and W. Samek, “Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, September 2020.

[20] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, “FedBN: Federated Learning on Non-IID Features via Local Batch Normalization,” in *Proc. ICLR 2021*, May 2021.

[21] 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*, Early Access, July 2021.

[22] S. Han, J. Pool, J. Tran, and W. Dally, “Learning Both Weights and Connections for Efficient Neural Network,” in *Proc. NIPS 2015*, December 2015.

[23] S. Anwar, K. Hwang, and W. Sung, “Structured Pruning of Deep Convolutional Neural Networks,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 3, pp. 32, February 2017.

[24] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision,” in *Proc. ICML 2015*, July 2015.

[25] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, “Training Deep Neural Networks with 8-bit Floating Point Numbers,” in *Proc. NIPS 2018*, December 2018.

[26] X. Sun, N. Wang, C. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. E. Maghraoui, V. Srinivasan, and K. Gopalakrishnan, "Ultra-Low Precision 4-bit Training of Deep Neural Networks," in *Proc. NIPS 2020*, vol. 33, 2020.

[27] E. Park and S. Yoo, "Profit: A Novel Training Method for Sub-4-Bit Mobilenet Models," in *Proc. ECCV 2020*, pp. 430–446, 2020.

[28] T. Liang, J. Glossner, L. Wang and S. Shi, "Pruning and Quantization for Deep Neural Network Acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[29] J. Mills, J. Hu, and G. Min, "Communication-Efficient Federated Learning for Wireless Edge Intelligence in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, July 2020.

[30] E. Mao, J.Ding, and V. Tarokh, "HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients," in *Proc. ICLR*, May 2021.

[31] D. Jhunjhunwala, A. Gadhikar, G. Joshi, and Y. C. Eldar, "Adaptive Quantization of Model Updates for Communication-Efficient Federated Learning," in *Proc. IEEE ICASSP 2021*, June 2021.

[32] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng. "TiFL: A Tier-Based Federated Learning System," in *Proc. HPDC 2020*, June 2020.

[33] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-FL for Wireless Networks: Cooperative Learning Mechanism Using Non-IID Data," in *Proc. IEEE ICC 2020*, June 2020.

[34] S. Caldas, S. Duddu, P. Wu, T. Li, J. Konecny, H. McMahan, V. Smith, and A. Talwalkar, "LEAF: A Benchmark for Federated Settings," arXiv preprint arXiv:1812.01097v3, December 2019.

[35] T. Zhang, Z. Lin, G. Yang, and C. De Sa, "QPyTorch: A Low-Precision Arithmetic Simulation Framework," arXiv preprint arXiv:1910.04540, October 2019.

[36] SITUA-CQ, [Online].Available: https://github.com/wwtkddnjsww/SITUA-CQ

[37] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009.

[38] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian Nonparametric Federated Learning of Neural Networks," in *Proc. ICML*, June 2019.

[39] Microsoft DeepSpeed, [Online]. Available: https://www.deepspeed.ai

[40] C. -H. Wang, K. -Y. Huang, J. -C. Chen, H. -H. Shuai, and W. -H. Cheng, "Heterogeneous Federated Learning Through Multi-Branch Network," in *Proc. IEEE ICME*, July 2021.

[41] Y. Yang, L. Deng, S. Wu, T. Yan, Y. Xie, and G. Li, "Training High-Performance and Large-Scale Deep Neural Networks with Full 8-Bit Integers," *Neural Networks*, vol. 125, pp. 70—82, May 2020.

Sangwon Seo received the B.S. degree from the School of Computer Engineering, Chungbuk National University, Cheongju, Korea, in 2020. He received M.S degree from the School of Electrical Engineering, Korea University, Seoul, Korea, in 2022. He is currently doctoral student at the Division of Information Science and Engineering, the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden. His research interests include 5G/6G networks, latency prediction, internet of things, and federated learning.

Jaewook Lee received the B.S. and Ph.D from the School of Electrical Engineering, Korea University, Seoul, Korea, in 2014 and 2021, respectively. He is currently an assistant professor in the Department of Information and Communication Engineering, Pukyong National University, Busan, Korea. From 2021 to 2023, He was an senior researcher at the Electronics and Telecommunications Research Institute, Daejon, Korea. His research interests include 6G mobile networks, federated learning, network automation and time sensitive networking.

Haneul Ko [M'19] received B.S. and Ph.D. from the School of Electrical Engineering, Korea University, Seoul, Korea, in 2011 and 2016, respectively. He is currently an assistant professor in the Department of Electronic Engineering, Kyung Hee University, Yongin, Korea. From 2019 to 2022, he was an assistant professor in the Department of Computer and Information Science, Korea University, Sejong, Korea. From 2017 to 2018, he was a Postdoctoral Fellow at the University of British Columbia, Vancouver, BC, Canada. From 2016 to 2017, he was a Postdoctoral Fellow in mobile network and communications, Korea University, Seoul, Korea. He was the recipient of Minister of Education Award in 2019 and IEEE ComSoc APB Outstanding Young Researcher Award in 2022. His research interests include 5G/6G networks, network automation, mobile cloud computing, SDN/NFV, and Future Internet.

Sangheon Pack [SM'11] received B.S. and Ph.D. degrees from Seoul National University, Seoul, Korea, in 2000 and 2005, respectively, both in computer engineering. In 2007, he joined the faculty of Korea University, Seoul, Korea, where he is currently a Professor in the School of Electrical Engineering. From 2005 to 2006, he was a Postdoctoral Fellow with the Broadband Communications Research Group, University of Waterloo, Waterloo, ON, Canada. He was the recipient of the IEEE/Institute of Electronics and Information Engineers (IEIE) Joint Award for IT Young Engineers Award 2017, Korean Institute of Information Scientists and Engineers (KIISE) Young Information Scientist Award 2017, Korea University TechnoComplex (KUTC) Crimson Professor 2015, Korean Institute of Communications and Information Sciences (KICS) Haedong Young Scholar Award 2013, LG Yonam Foundation Overseas Research Professor Program in 2012, and IEEE ComSoc APB Outstanding Young Researcher Award in 2009. He served as a TPC vice-chair for information systems of IEEE WCNC 2020, a track chair of IEEE VTC 2020-Fall/2010-Fall and IEEE CCNC 2019, a TPC chair of IEEE/IEIE ICCE-Asia 2018/2020, EAI Qshine 2016, and ICOIN 2020, a publication co-chair of IEEE INFOCOM 2014 and ACM MobiHoc 2015, a symposium chair of IEEE WCSP 2013, a TPC vice-chair of ICOIN 2013, and a publicity co-chair of IEEE SECON 2012. He is an editor of IEEE Internet of Things (IoT) Journal, Journal of Communications Networks (JCN), IET Communications, and he was a guest editor of IEEE Transactions on Emerging Topics in Computing (TETC) and IEEE Transactions on Network Science and Engineering (TNSE). He is a senior member of IEEE. His research interests include softwarized networking (SDN/NFV), 5G/6G mobile core networks, mobile edge computing/programmable data plane, and vehicular networking.