This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

# Virtual Experience-Based Mobile Device Selection Algorithm for Federated Learning

Mincheol Paik, Haneul Ko, *Member, IEEE*, and Sangheon Pack, *Senior Member, IEEE*

***Abstract*—In federated learning (FL), to minimize the convergence time while reaching a desired accuracy, it is important to select appropriate mobile devices (MDs) as participants with the considerations of their characteristics (e.g., data distribution, channel condition, and computing power). In this article, we first formulate a Markov decision process-based MD selection problem in which the increased accuracy per unit-time is maximized. To solve the formulated problem without any prior knowledge of the environment, a deep Q-network (DQN) algorithm can be exploited. However, storing previous experiences into the replay memory for DQN consumes an increased time since the FL server needs to conduct a number of actual FL procedures and observe the resulted rewards. To address this problem, we proposed a virtual experience-based MD selection algorithm (VE-MSA). In VE-MSA, the FL server generates virtual experiences (especially reward) without any actual FL procedures by using two neural networks approximating the round time and the increased accuracy in a round according to the selected MDs, respectively. Evaluation results demonstrate that the derived optimal policy can achieve a target accuracy within the shortest time among comparison schemes.**

***Index Terms*—Client selection, deep Q-network (DQN), experience relay, federated learning (FL), straggler problem.**

## I. Introduction

TRADITIONALLY, huge data are collected from different locations and examined in the centralized cloud to train various deep learning models [1], [2]. This centralized method has the following issues [3]. First, the owner having the privacy data does not want to share it with the cloud. Moreover, the centralized method can lead undesirable latency for real-time applications, e.g., augmented reality. In addition, the data transmission can cause severe congestion in core networks due to huge volume of data.

To alleviate these issues, lots of researchers have investigated decentralized model generation methods such as federated learning (FL), where a model (e.g., image classification model) is collaboratively made by distributed agents [4]. In FL, the learning is conducted based on a round. The round proceeds according to the following steps. First, an FL server randomly selects some participants.[1] Then, the FL server transmits a deep learning model with initial parameters to the selected participants. After receiving them, the participants train the model by using their own data, and then return newly trained parameters to the FL server. Then, the FL server aggregates the parameters from the participants and repeats these steps (i.e., a round) until achieving a desired performance. In so doing, we can make the models without the issues on the data privacy, undesirable latency, and congestion of core networks.

Since the performance of mobile devices (MDs) is improved, they can be considered as participants in FL [3]. However, each MD has different characteristics such as data distribution, channel condition, and computing power. Therefore, MDs have different impacts or contributions to the minimization of the model convergence time for achieving a desired accuracy. For example, if MDs having poor channel condition and/or low computing power are chosen as participants, they cannot return newly trained parameters to the FL server in a timely manner (i.e., they have a limited contribution). In addition, if the aggregated data distribution of chosen MDs is quite different from the global data distribution, the model accuracy cannot be effectively increased even with the increase of the number of MDs [6]. Consequently, it is vital to choose the most appropriate MDs as participants in FL. To this end, serveral previous works exploited a deep Q-network (DQN) algorithm; however, storing previous experiences into the replay memory for DQN consumes an increased time, since the FL server needs to conduct a number of actual FL procedures and observe the resulted rewards.

In this article, we investigate a MD selection policy in FL. For this, we formulate a MD selection problem based on Markov decision process (MDP) with the objective to select appropriate MDs who can contribute to maximize the increased accuracy per unit-time (i.e., minimize the convergence time). To solve the formulated problem without a prior knowledge of the environment, the DQN algorithm [7] can be exploited. However, to store experiences into the replay memory for DQN, the agent (i.e., FL server) conducts the actual FL procedure. Then, only

Manuscript received 18 January 2022; revised 15 July 2022; accepted 9 September 2022. This work was supported in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korea government (MSIT) under Grant 2022-0-01015, in part by the Development of Candidate Element Technology for Intelligent 6G Mobile Core Network and in part by National Research Foundation (NRF) of Korea Grant funded by the Korean Government (MSIP) under Grant 2021R1A4A3022102. *(Corresponding authors: Haneul Ko; Sangheon Pack.)*

Mincheol Paik is with the Department of Computer and Information Science, Korea University, Sejong 30019, South Korea (e-mail: fall123123@korea.ac.kr).

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

Sangheon Pack is with the School of Electrical Engineering, Korea University, Seoul 02841, South Korea (e-mail: shpack@korea.ac.kr).

Digital Object Identifier 10.1109/JSYST.2022.3206404

[1]To encourage the participation, the FL server can provide some incentives (e.g., monetary reward) [5].

1937-9234 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

after all selected MDs return their updated parameters to the FL server, the reward (i.e., increased accuracy per unit-time) can be observed, which consumes a long time. To address this problem, we introduce a virtual experience-based MD selection algorithm (VE-MSA). In VE-MSA, the FL server trains two neural networks that approximate a round time and an increased accuracy in a round according to the selected MDs, respectively. These neural networks can catch how much characteristics of MDs (e.g., data distribution, channel condition, computing power, etc.) influence the round time and model accuracy, and therefore these two neural networks can generate a reward signal (i.e., increased accuracy per unit-time) for DQN. Note that these two neural networks can achieve high accuracy regardless of training data types of FL (see Table II), which implies that these networks are reusable to train various models by FL. Evaluation results demonstrate that the derived optimal policy can achieve a target accuracy within the shortest time among comparison schemes. In addition, it can be shown that the proposed algorithm can reduce the time of obtaining experiences for the replay memory.

The contribution of this article can be summarized as follows: 1) the developed VE-MSA allows to select appropriate MDs, who can maximally contribute to the convergence time minimization based on DQN algorithm; 2) the introduced two neural networks catch the relationship between MD characteristics and the model accuracy, which can be used as a reward signal (i.e., increased accuracy per unit-time) for DQN. As a result, DQN can generate virtual experiences without any actual FL procedure; and 3) we conduct extensive evaluation under various environments, which can be valuable guidelines for selecting MDs in FL.

The remainder of this article is organized as follows. The related work is summarized in Section II. The system model is described in Section III. The reinforcement learning problem is formulated in Section IV, and the learning method is presented in Section V. The evaluation results are demonstrated in Section VI. Finally, Section VII concludes this article.

## II. Related Work

In FL, each MD has different characteristics and these characteristics affect the training efficiency. Therefore, it is important which MDs are selected, and related studies have been conducted in the literature [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22].

Nishio and Yonetani [8] introduced a MD selection algorithm that determines a time deadline and selects only MDs who are able to complete the model update within that deadline. Jin et al. [9] formulated a nonlinear integer programming problem on the MD selection to minimize the total resources of MDs for FL. However, in [8] and [9], the data distribution of MDs was not considered when selecting MDs. Yoshida et al. [10] proposed a learning mechanism, where the FL server requests the data uploading to few MDs for constructing global independent and identically distributed (IID) dataset and updates the model by using this dataset to improve the training efficiency. Kang et al. [11] introduced a MD selection algorithm that selects only high-reputation MDs to eliminate unreliable local model update. However, they did not provide any detailed optimization method. Yang et al. [12] conducted a joint device selection and beamforming design to increase the number of selected MDs while satisfying the mean-square-error (MSE) requirement. Xia et al. [13] proposed a reinforcement learning-based MD scheduling algorithm to maximize the training efficiency. Similarly, Yoshida et al. [15] introduced a reinforcement learning-based MD selection algorithm that considers the performance of MDs such as the computing resource and channel quality to avoid selecting straggler. Ko et al. [16] formulated a joint optimization problem on MD selection and bandwidth allocation to minimize the convergence time for a desired accuracy. However, the presented MD selection method requires prior environmental knowledge (i.e., transition probability). Yang et al. [14] investigated an elapsed time-based MD scheduling algorithm, where MDs who updated parameters for a long time ago are selected. Huang et al. [17] investigated the tradeoff between fairness and training efficiency, and then designed a fairness guaranteed selection algorithm to obtain the solution of the formulated problem with a polynomial complexity. Zhang et al. [18] proposed a MD selection algorithm that selects MDs with lower degree of non-IID data and higher frequency of participating model training. Zhang et al. [19] proposed a deep Q-learning (DQL)-based MD selection policy considering channel state, computing power, and energy state of MDs to solve the latency and energy over-consumption problems in FL. Ma et al. [20] proposed a FL mechanism, in which the FL server aggregates parameters from a specific number of MDs in each round to minimize the convergence time for a desired accuracy. Yu et al. [21] presented a heuristic approach to solve a joint optimization problem on MD selection and resource management to maximize the number of aggregated models in each round while minimizing the total energy consumption of MDs. Zhang et al. [18] proposed a MD selection algorithm to select MDs having near-IID data by defining a weight divergence between MDs. Tang et al. [22] proposed a correlation-based MD selection algorithm that selects MDs having strong data correlation with other MDs in order to minimize the total loss of the model.

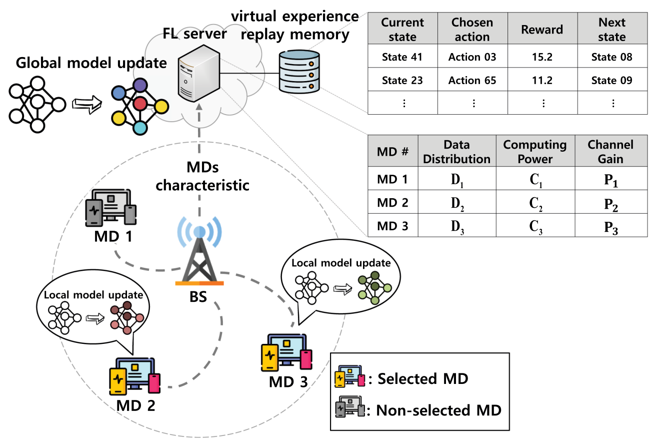
However, there is no previous work to exploit neural networks to determine the effect of MD characteristics on the model accuracy. Note that, in this article, we introduce neural networks that can generate the reward signal reflecting the characteristics of MDs on the model accuracy.

## III. System Model

Fig. 1 illustrates the system model in this article. It is assumed that $N$ MDs exist and the FL server is located at base station (BS). The FL server wants to train a model of $W$ MByte [8] in a distributed manner. Note that, we assume that BSs are interconnected through the Xn interface, and thus the updated models can be always delivered to the FL server even with the mobility of MDs. For example, when MDs move out the coverage of BS, where the FL server is located, they can send the updated models to the FL server through the Xn interface. Therefore, how to control mobility of MDs was not considered in this work.

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



| Current state | Chosen action | Reward | Next state |
|---|---|---|---|
| State 41 | Action 03 | 15.2 | State 08 |
| State 23 | Action 65 | 11.2 | State 09 |
| ⋮ | ⋮ | ⋮ | ⋮ |

| MD # | Data Distribution | Computing Power | Channel Gain |
|---|---|---|---|
| MD 1 | $D_1$ | $C_1$ | $P_1$ |
| MD 2 | $D_2$ | $C_2$ | $P_2$ |
| MD 3 | $D_3$ | $C_3$ | $P_3$ |

Fig. 1. System model.

To select MDs participating into FL, the FL server periodically collects some characteristics of MDs such as the data distribution $D$, channel gain $C$, and available computing power $P$ (in units of floating point operations per second (FLOPS) [23]). These characteristics affect the convergence time that a desired accuracy of the model is achieved. Specifically, if the data distribution $D$ of a specific MD is more close to the global data distribution $D_G$,[2] a short convergence time is expected [25]. In addition, more available computing power $P$ of MDs indicates that they can finish their local training within shorter duration (i.e., shorter convergence time). Moreover, if MDs have better channel gain $C$, they can transmit the updated model with faster transmission rate, which leads shorter convergence time.

By considering these characteristics (i.e., data distribution, channel gain, and available computing power), the FL server selects MDs who can contribute to maximize the increased accuracy per unit-time (i.e., minimize the convergence time) based on Q-network of which inputs are the characteristics of MDs.

To train Q-network (i.e., to use DQN algorithm), the FL server should store sufficient experiences[3] into the replay memory [7]. However, it is difficult to mathematically define and/or calculate the reward (i.e., increased accuracy per unit-time according to the selected MDs). Thus, to store experiences into the replay memory for DQN algorithm [7], the agent (i.e., FL server) conducts the actual FL procedure. Then, only after all selected MDs upload their updated model to the FL server, the FL server can observes the reward (i.e., increased accuracy per unit-time), which consumes a long time. To address this problem, we introduce VE-MSA, where the FL server trains two neural networks [called time neural network (TNN) and accuracy neural network (ANN), respectively] of which outputs are the round time and the increased accuracy (see Fig. 2). For training these neural networks, the FL server observes the round time and the increased accuracy which are used as labels of TNN and ANN, respectively. Meanwhile, the round time depends on the slowest MD (i.e., straggler), whereas the increased accuracy is decided by the current model accuracy and the aggregated data distribution of selected MDs. Therefore, the computing power and channel gain of the straggler are used as inputs of TNN. In addition, the current model accuracy and the aggregated data distribution of selected MDs are exploited as inputs of ANN. After finishing the training, these neural networks can be used to generate a reward signal (i.e., to calculate the increased accuracy per unit-time according to the selected MDs) for Q-network. Specifically, the reward can be obtained by dividing the output of ANN by that of TNN. Consequently, experiences consisting of the current state, action, reward, and next state can be obtained without any actual FL procedure.

ANN and TNN can achieve high accuracy regardless of training data types of FL (see Table II), because the outputs of these two neural networks (i.e., round time and increased accuracy) are not influenced by training data types of FL (see inputs of TNN and ANN in Fig. 2). Therefore, once the FL server trains TNN and ANN, it can use pretrained TNN and ANN as criteria of selecting MDs.[4] That is, the FL server can store virtual experiences into the replay memory without any actual FL procedure and derive the optimal policy on the MD selection within a short duration.[5]

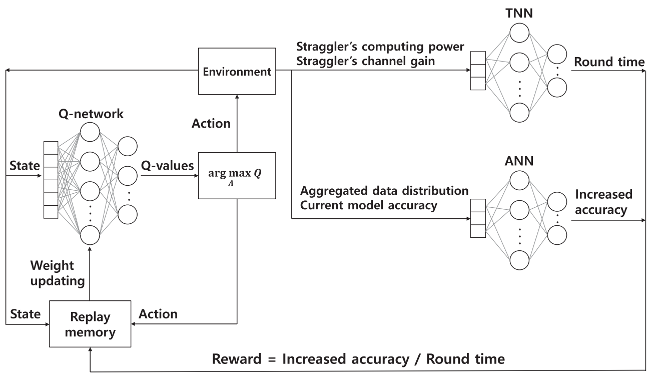


Fig. 2. Interaction among neural networks.

## IV. Reinforcement Learning Problem

To select appropriate MDs, who can contribute to maximize the increased accuracy per unit-time (i.e., minimize the convergence time), we formulate MDP model, where the agent (i.e., FL server) periodically conducts appropriate actions to maximize (or minimize) the reward (or cost). In other words, in this article, the FL server selects appropriate MDs at time

[2]To evaluate the similarity between the data distribution $D$ of MDs and the global data distribution $D_G$, the cosine similarity is exploited [24]. Specifically, the cosine similarity between $D$ and $D_G$ can be calculated as $\frac{D_G \circ D}{\|D_G\|\|D\|}$, where $\circ$ is the Hadamard product and $||D||$ is the Euclidean norm of the distribution $D$. Note that, when the data class is discrete, the data distribution can be considered as a kind of vectors and thus its Euclidean norm can be easily calculated.

[3]The one experience consists of the current state, the chosen action, reward, and the next state.

[4]Note that TNN and ANN can achieve a sufficient accuracy regardless of training data type, which will be demonstrated in Section VI (see Table II).

[5]If the virtual experiences are exposed to the attackers, they can estimate the characteristics of MDs (e.g., data distribution, channel condition, and computing power). To avoid this situation, the FL server can adopt various encryption algorithms such as SEED and SHA-256.

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
SUMMARY OF NOTATIONS

| Notation | Description |
|---|---|
| $\mathbf{S}$ | Overall state space |
| $\mathbf{G}$ | State space for the accuracy of the global model |
| $\mathbf{D_i}$ | State space for the data distribution of MD $i$ |
| $\mathbf{C_i}$ | State space for the channel of MD $i$ |
| $\mathbf{P_i}$ | State space for the computing power of MD $i$ |
| $g_u$ | Unit of accuracy |
| $g_{\max}$ | Maximum accuracy |
| $D_i^k$ | Number of data for class $k$ of MD $i$ |
| $D_{\max}$ | Maximum number of data for class $k$ |
| $C_i^1$ | Lowest channel gain of MD $i$ |
| $C_i^{\max}$ | Highest channel gain of MD $i$ |
| $P_i^1$ | Minimum computing power of MD $i$ |
| $P_i^{\max}$ | Maximum computing power of MD $i$ |
| $\mathbf{A}$ | Overall action space |
| $\mathbf{A_i}$ | Action space for MD $i$ |
| $R$ | Reward function |
| $T_R$ | Round time |
| $A_I$ | Increased accuracy during a round |

epoch $T = \{1, 2, 3, \ldots\}$. Important notations are summarized in Table I.

### A. State Space

The state space $\mathbf{S}$ can be defined as

$$\mathbf{S} = \mathbf{G} \times \prod_{\mathbf{i}} \mathbf{D_i} \times \mathbf{C_i} \times \mathbf{P_i} \tag{1}$$

where $\mathbf{G}$ is the state space for the accuracy of the global model. In addition, $\mathbf{D_i}$ and $\mathbf{C_i}$ denote the distribution state space and the channel state space of MD $i$, respectively. $\mathbf{P_i}$ represents the computing power state space of MD $i$.

$\mathbf{G}$ can be represented as

$$\mathbf{G} = \{0, g_u, 2g_u, \ldots, g_{\max}\} \tag{2}$$

where $g_u$ is a unit of accuracy and $g_{\max}$ is the maximum accuracy (e.g., 100%).

$\mathbf{D_i}$ can be represented as

$$\mathbf{D_i} = \prod_k D_i^k \tag{3}$$

where $D_i^k$ is the number of data for class $k$ of MD $i$, which can be described by

$$D_i^k = \{0, 1, 2, \ldots, D_{\max}\} \tag{4}$$

where $D_{\max}$ denotes the maximum number of data for class $k$.

When the channel gain model is discretized,[6] $\mathbf{C_i}$ can be defined as

$$\mathbf{C_i} = \left\{C_i^1, C_i^2, \ldots, C_i^{\max}\right\} \tag{5}$$

where $C_i^1$ and $C_i^{\max}$ denote the lowest and highest channel gains of MD $i$, respectively.

[6]For example, in the LTE system, the discrete channel quality indicator is used by MDs to indicate the channel quality to BS.

$\mathbf{P_i}$ can be described by

$$\mathbf{P_i} = \left\{P_i^1, P_i^2, \ldots, P_i^{\max}\right\} \tag{6}$$

where $P_i^1$ and $P_i^{\max}$ denote the minimum and maximum computing powers of MD $i$, respectively.

### B. Action Space

The action space $\mathbf{A}$ can be defined by

$$\mathbf{A} = \prod_{\mathbf{i}} \mathbf{A_i} \tag{7}$$

where $\mathbf{A_i}$ is the action space for MD $i$.

$\mathbf{A_i}$ can be represented as

$$\mathbf{A_i} = \{0, 1\} \tag{8}$$

where $A_i$ denotes whether MD $i$ is selected or not. That is, if $A_i = 0$, MD $i$ is not selected. Otherwise, MD $i$ is selected.

### C. Reward

The objective of the agent is to maximize the increased accuracy per unit-time. Therefore, the increased accuracy per unit-time should be a reward function $R$, which can be represented by

$$R = \frac{A_I}{T_R} \tag{9}$$

where $T_R$ and $A_I$ denote the round time and the increased accuracy during the round time, respectively. However, since it is difficult to mathematically obtain $T_R$ and $A_I$, we exploit TNN and ANN of which outputs are $T_R$ and $A_I$, respectively, as described in Section III. $R$ can be obtained by dividing the output of ANN by that of TNN.

When an environment (i.e., transition probability) is given, the formulated MDP model can be solved by a value iteration algorithm. Each iteration of the algorithm can be conducted with $O(|\mathbf{A}||\mathbf{S}|^2)$. However, the transition probability in our problem cannot be easily obtained, and thus we cannot exploit this value iteration algorithm. Instead of that, we have utilized the DQN algorithm, which can operate without any information on the transition probability.

## V. Learning Method

To apply DQN, the optimal action-value function $Q^*(S, A)$ can be defined by [7]

$$Q^*(S, A) = \max_\pi E\left[R_t + \sum_i \gamma^i R_{t+i} | S_t, A_t, \pi\right] \tag{10}$$

where $\pi$ is a policy mapping states to actions, and $\gamma$ is the discount factor to control the amount of weight to future prediction. In addition, $S_t$ and $A_t$ are the state and the chosen action at the decision epoch $t$, respectively. $R_t$ is the reward received at $t$.

A deep neural network is utilized to approximate an appropriate action-value function $Q(S, A, \theta)$ with weights $\theta$. That is, the deep neural network inputs the state information and outputs Q-values for each of the possible actions. The deep

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

neural network is trained by adjusting weights $\theta$ at each iteration to minimize the loss functions, where the loss function at the $i$th iteration is defined as [7]

$$L_i(\theta_i) = E\left[(y - Q(S_t, A_t, \theta_i))^2\right] \tag{11}$$

where $y$ is the target value, which can be defined as (12) shown at the bottom of this page.

To prevent that Q-network fails to converge due to a moving target, we utilize the target network separation as similar in [7]. Meanwhile, the concept of the experience replay memory [7], where the agent conducts an action and stores the experience $(S_t, A_t, R_t, S_{t+1})$, is used to break the correlation between successive experiences and learn the underlying distribution of labels in the model. However, in our problem, it takes too long time to store one experience, because the agent (i.e., FL server) can know the reward $R_t$ (i.e., increased accuracy per unit-time for the global model) after all selected MDs finish their training and return the newly updated model to the FL server. To mitigate this problem, we design the concept of the virtual experience replay memory $M$. Specifically, based on TNN and ANN, the agent can calculate the reward $R_t$ without waiting that all selected MDs return the updated model. Then, the agent can store the virtual experience $(S_t, A_t, R_t, S_{t+1})$ at the virtual experience replay memory.

The detailed DQN with the virtual experience replay memory is presented in Algorithm 1. First, the algorithm initializes the virtual experience replay memory $M$, the main DQN, and the target DQN (see lines 1–3 in Algorithm 1). After that, the algorithm trains the deep neural network as follows. First, the controller observes the initial state $S_t$ (line 5 in Algorithm 1) and selects the action $A_t$ according to the $\epsilon$-greedy method (lines 7–12 in Algorithm 1). After that, it virtually executes the selected action $A_t$ and calculates the reward $R_t$ based on TNN and ANN (line 13 in Algorithm 1). Then, the algorithm stores virtual experience $(S_t, A_t, R_t, S_{t+1})$ into the virtual experience replay memory $M$ and randomly selects a set of experiences from that memory (lines 14–15 in Algorithm 1). Based on these selected experiences, the algorithm sets the target value $y_i$ by using the target DQN (line 16 in Algorithm 1), and trains the main DQN by means of the loss function and the gradient descent method (line 17 in Algorithm 1). Then, the target DQN is updated periodically by copying the weights from those of the main DQN (line 18 in Algorithm 1).

**Algorithm 1:** DQN With the Virtual Experience Replay Memory.

```
1:  Initialize the virtual experience replay memory $M$
2:  Initialize the main deep Q-network with weights $\theta$
3:  Initialize the target deep Q-network with weights
    $\overline{\theta} = \theta$
4:  for each episode $j = 1, 2, 3, \ldots J$ do
5:    Receive the initial observation $S_t$
6:    for $t = 1, 2, 3, .., T$ do
7:      Choose a random probability $p$
8:      if $p \leq \epsilon$ then
9:        Randomly select an action $A_t$
10:     else
11:       $A_t = \arg\max_A Q(S_t, A, \theta)$
12:     end if
13:     Virtually execute the selected action $A_t$ and
        obtains the reward $R_t$ based on TNN and ANN
        and the next state $S_{t+1}$
14:     Store virtual experience $(S_t, A_t, R_t, S_{t+1})$ into the
        virtual experience $M$
15:     Get a set of virtual experiences $(S_i, A_i, R_i, S_{i+1})$
        into the virtual experience $M$
16:     Set
        $y_i = \begin{cases} R_i, \text{ if episode terminates at the next state} \\ R_i + \gamma \max_{A'} Q(S_{i+1}, A,' \overline{\theta}), \text{ otherwise} \end{cases}$
17:     Performs a gradient descent step on
        $(y_i - Q(S_i, A_i, \theta))^2$ with respect to $\theta$
18:     For every $K$ steps, update the target deep
        Q-network with $\overline{\theta} = \theta$
19:   end for
20: end for
```

## VI. Evaluation Results

For the performance evaluation, we compare VE-MSA with three schemes: 1) FAST [8] in which the FL server selects the fastest MDs without the consideration of their distribution; 2) DIST [26] in which the FL server selects MDs to make their aggregated data distribution similar to the global data distribution; and 3) ORIG [4] in which the FL server randomly selects MDs. For the fair comparison, the number of selected MDs is set to three in all schemes. Meanwhile, for the performance metrics, the convergence time to achieve a target accuracy, the number of needed rounds to achieve a target accuracy, and the average round time are measured.

Using Python, we have developed a FL simulator, where multiple MDs having different characteristics (i.e., data distribution, channel condition, and computing power) are generated for each simulation run. We have conducted $10^3$ simulation runs with different seed values for each point and obtained the average value. As similar to the previous FL studies [4], [8] CIFAR-10, which is one of classification dataset, is exploited. The other default simulation parameters are as follows. The number of total MDs is 10. The parameter size $W$ of the model is 18 MByte. The bandwidth and transmission power are set to 3 MHz and 7 dBM/MHz, respectively [27]. In addition, noise power is set to 1 dBM/MHz. The channel gains and available computing powers

$$y = \begin{cases} R_t, & \text{if episode terminates at the next state} \\ R_t + \gamma \max_{A'} Q(S_{t+1}, A,' \theta_{i-1}), & \text{otherwise} \end{cases} \tag{12}$$

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
ROOT MEAN SQUARE ERROR (RMSE) OF A REWARD SIGNAL

| Dataset | RMSE |
|---|---|
| CIFAR-10 | 0.085 |
| CIFAR-100 | 0.138 |
| MNIST | 0.134 |

TABLE III
EXPERIENCE TIME (UNIT: SEC)

| Number of experiences | W/O TNN & ANN | With TNN & ANN |
|---|---|---|
| 5000 | 7831.2 | 6.803 |
| 10000 | 16332.0 | 21.032 |
| 15000 | 18538.8 | 45.300 |
| 20000 | 26767.2 | 98.489 |

of MDs are normalized to values between 0 and 1. The number of data in MD is set to 500 [28]. The minimum and maximum available computing power are set to 0.3 and 1.0 MHz. Note that, since we have exploited a small DQN for our evaluations (i.e., DQN employed has only three hidden layers and each layer consists of 128 neurons), the complexity of training DQN is not quite high.

## A. Accuracy of Pretrained TNN and ANN

In our simulation, we train TNN and ANN based on CIFAR-10 dataset. To test the accuracy of these two networks, we divide CIFAR-10 dataset into training and test data sets, and we calculate a root mean square error (RMSE) of a reward signal for the test dataset. As shown in Table II, RMSE of a reward signal for the CIFAR-10 test dataset is 0.085 (i.e., TNN and ANN can generate an accurate reward signal). In addition, from Table II, it can be found that these two neural networks can achieve high accuracy regardless of data types (i.e., RMSEs for CIFAR-100 and MNIST are maintained as low level, 0.138 and 0.134, respectively). This is because the outputs of these two neural networks (i.e., round time and increased accuracy) are not influenced by data types (see inputs of TNN and ANN in Fig. 2). Based on this result, when the FL server has the pretrained TNN and ANN, it can store virtual experiences into the virtual experience replay memory without any actual FL procedure, which can significantly reduce the learning time of Q-network.

## B. Effect of the Virtual Experience Replay Memory

Table III shows the total experience times according to the number of experiences. As shown in Table III, it can be found that the total experience time without TNN and ANN is much longer than that with TNN and ANN. This is because the actual FL procedure should be conducted when the agent does not use TNN and ANN. That is, without TNN and ANN, only after all selected MDs upload their updated model to the FL server, the FL server can observe the increased accuracy per unit-time. On the other hand, with TNN and ANN, since the agent just conducts the inference, it can gain lots of experiences in a short time, which leads the significant reduction of the learning time of Q-network.

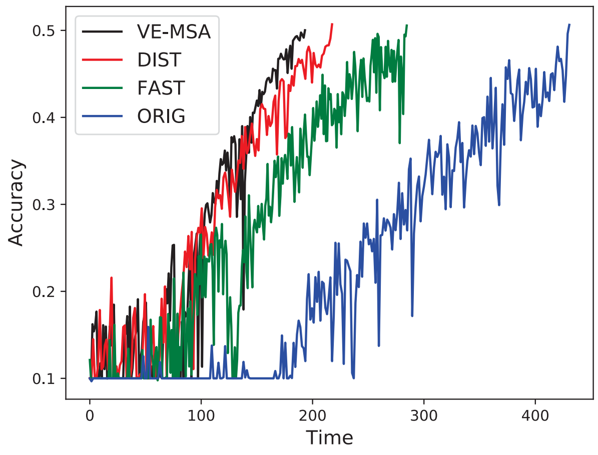


Fig. 3. Accuracy curve of different schemes.

## C. Learning Accuracy

Fig. 3 shows the learning accuracy as time goes. As shown in Fig. 3, the accuracy of VE-MSA increases faster than other schemes. This can be explained as follows. First, VE-MSA selects MDs having data distribution close to the global data distribution, which allows sufficient accuracy improvement in each round. In addition, since VE-MSA selects MDs with better channel quality and richer computing power, each round in VE-MSA can be completed faster than other schemes.

## D. Effect of Cosine Similarity

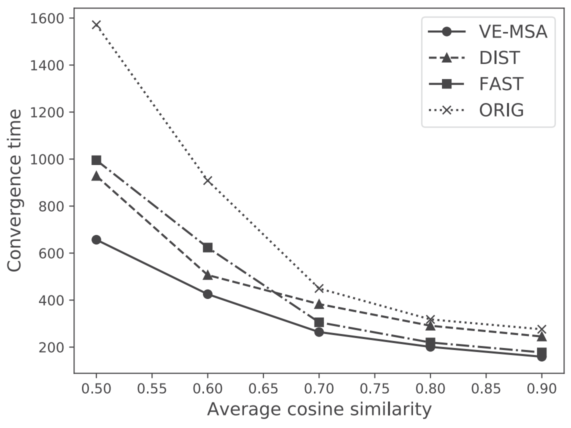
Fig. 4 shows the effect of the average cosine similarity of MDs. From Fig. 4(a), it can be shown that the convergence time of VE-MSA to achieve a target accuracy is shorter than that of other schemes. This is because VE-MSA selects MDs having data distribution similar to the global data distribution, better channel gain, and more available computing power. Therefore, MDs can guarantee sufficient improvement of accuracy in each round. Moreover, they can transmit the updated models within shorter duration. From Fig. 4(a), it can be found that the convergence time to achieve a target accuracy of all schemes decreases as the average cosine similarity of MDs increases. This can be explained as follows. When MDs have similar data distribution with the global data distribution (i.e., high cosine similarity), the updated parameters from MDs have similar directions. In this situation, the parameters from different MDs can be converged with fewer aggregation (i.e., fewer number of needed rounds), which can be observed in Fig. 4(b).

Meanwhile, because DIST selects MDs to make their aggregated data distribution similar to the global data distribution, its number of needed rounds is similar with that of VE-MSA [see Fig. 4(b)]. However, DIST does not consider the performance of MDs such as the channel state and computing power, and therefore its average round time is higher than that of VE-MSA [see Fig. 4(c)]. As a result, the convergence time of DIST is longer than that of VE-MSA [see Fig. 4(a)].

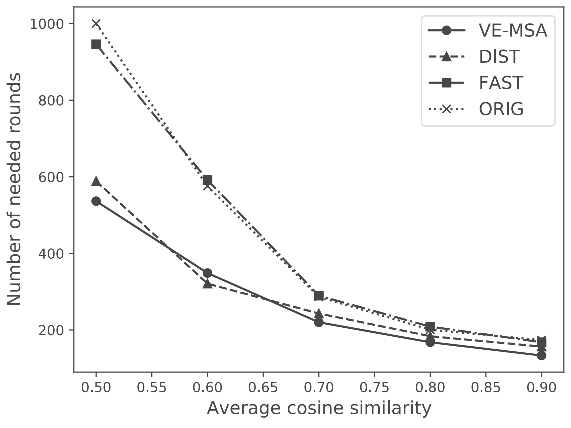
From Fig. 4(b), it can be found that the difference in the numbers of rounds between the data distribution-aware schemes (i.e., VE-MSA and DIST) and the data distribution nonaware

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

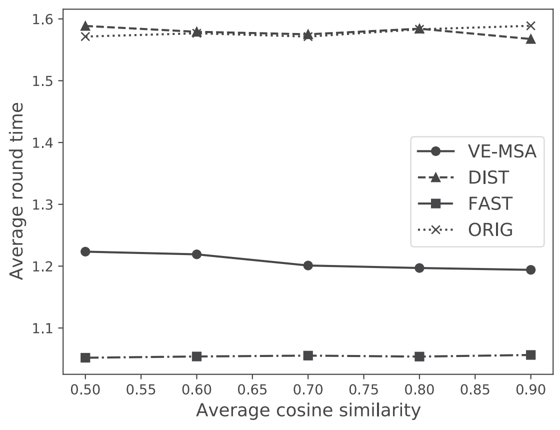
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



(a)

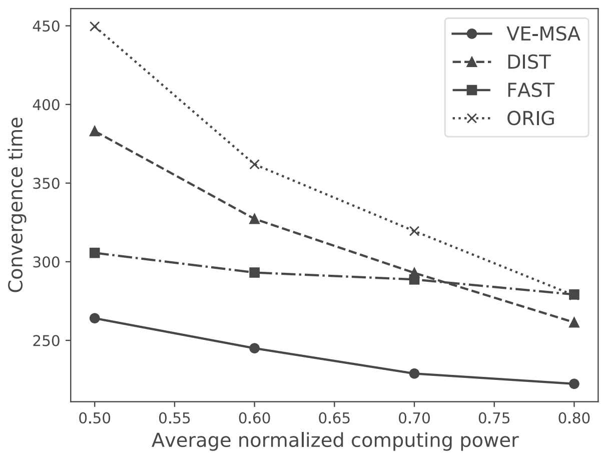


(b)

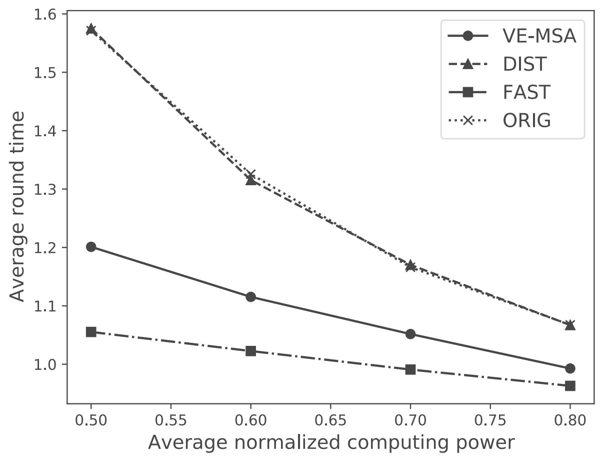


(c)

Fig. 4. Effect of the average cosine similarity of MDs. (a) Convergence time to achieve a target accuracy. (b) Number of needed rounds to achieve a target accuracy. (c) Average round time.



(a)



(b)

Fig. 5. Effect of the average normalized computing power of MDs. (a) Convergence time to achieve a target accuracy. (b) Average round time.

schemes (i.e., FAST and ORIG) is not distinct when the average cosine similarity of MDs is large. This can be explained as follows. In the situation, where the average cosine similarity of MDs is large, the aggregated data distribution of the selected MDs is always similar to the global data distribution even when MDs are randomly selected without consideration of their data distribution.

## *E. Effect of Average Computing Power of MDs*

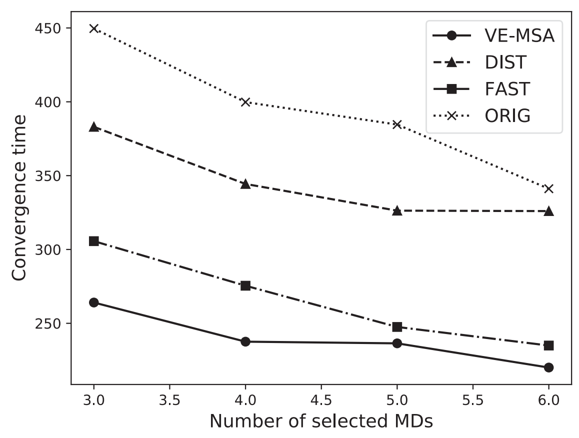
Fig. 5(a) and (b) shows the effect of the average normalized computing power[7] of MDs on the convergence time to achieve a target accuracy and the average round time, respectively. From Fig. 5(a), it can be found that the convergence time of all schemes decreases as the average normalized computing power increases. This is because selected MDs having higher computing power can finish the learning within shorter duration [i.e., can have shorter round time as shown in Fig. 5(b)].

Interestingly, as shown in Fig. 5(a), when the average normalized computing power is relatively small (i.e., $0.5 \sim 0.7$), the convergence time of DIST is larger than that of FAST. On the other hand, when the average normalized computing power is 0.8, the convergence time of FAST is larger than that of DIST. This can be explained as follows. When the average normalized computing power is relatively small (i.e., $0.5 \sim 0.7$), because DIST does not consider the performance of MDs, its average

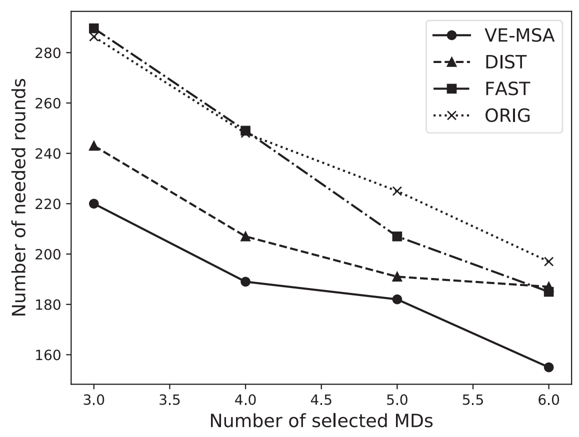
[7]The normalized computing power of a specific MD denotes the computing power divided by the average computing power of MDs.

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

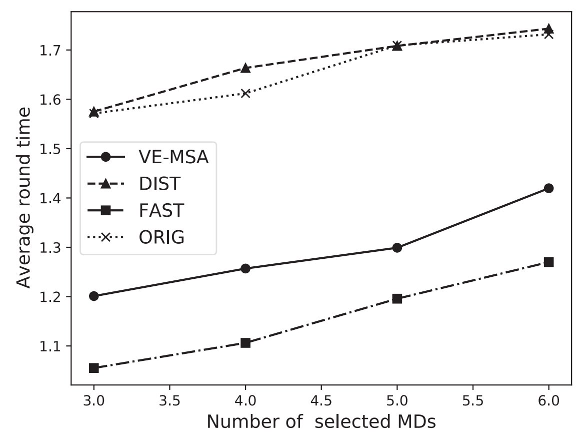
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



(a)

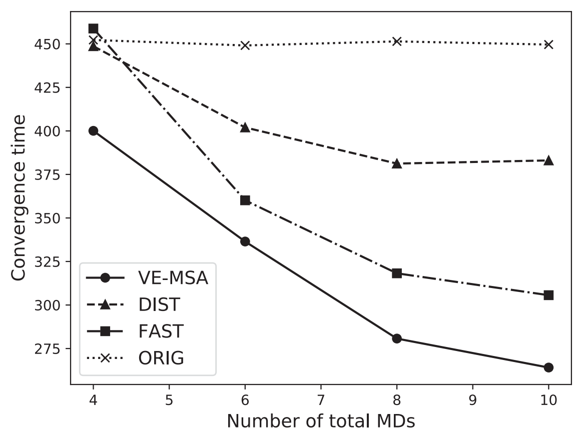


(b)

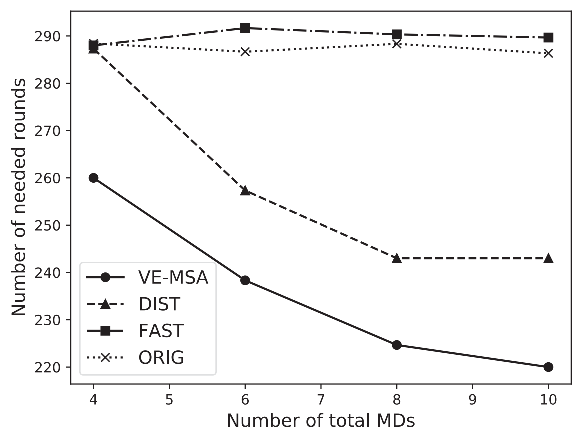


(c)

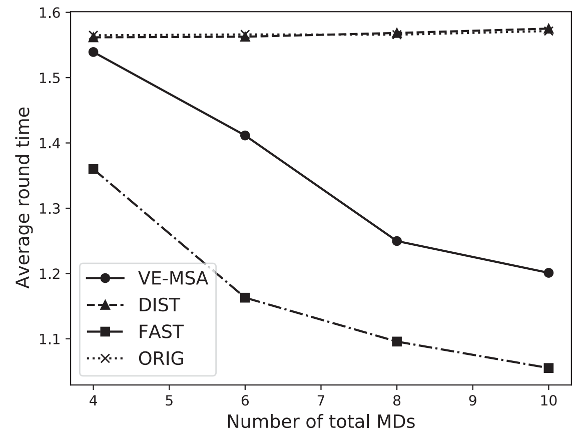
Fig. 6. Effect of the number of selected MDs. (a) Convergence time to achieve a target accuracy. (b) Number of needed rounds to achieve a target accuracy. (c) Average round time.



(a)



(b)



(c)

Fig. 7. Effect of the total number of candidate MDs. (a) Convergence time to achieve a target accuracy. (b) Number of needed rounds to achieve a target accuracy. (c) Average round time.

round time is much longer than that of FAST, which leads long convergence time. On the other hand, when the average normalized computing power is 0.8, even though MDs are selected randomly, most of them have enough computing power and thus they can finish the learning within a short duration (meaning short round time). To sum up, if the performance of MDs will be more improved, the data distribution of MDs becomes more important characteristic for the MD selection in FL.

### *F. Effect of the Number of Selected MDs*

Fig. 6 shows the convergence time to achieve a target accuracy, the number of needed round to achieve a target accuracy, and the

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

average round time as a function of the number of selected MDs in a round. As shown in Fig. 6(c), the average round time of all schemes increases as the number of selected MDs increases. This is because large number of selected MDs implies that MDs having worse performance (i.e., worse channel gain and computing power) can participate in the learning. However, as the number of MDs participating in learning increases, the amount of data used in the learning increases, resulting in an increase of the cosine similarity (i.e., aggregated data distribution in a single round is more similar to the global data distribution). Therefore, the number of needed rounds to achieve a target accuracy decreases [see Fig. 6(b)], which leads a shorter convergence time to to achieve a target accuracy [see Fig. 6(a)].

### G. *Effect of the Total Number of Candidate MDs*

Fig. 7(a) shows the effect of the total number of candidate MDs on the convergence time to achieve a target accuracy. From Fig. 7(a), it can be found that the convergence time of all schemes except ORIG decreases with the increase of the total number of candidate MDs. This is because all schemes except ORIG can select better MDs based on their criteria (e.g., data distribution, channel gain, and computing power) when there are lots of candidate MDs. Specifically, when MDs are selected by considering the data distribution (as in VE-MSA and DIST), the number of needed rounds to achieve a target accuracy in these schemes can be reduced [see Fig. 7(b)]. In addition, if the performance of MDs such as the channel state and computing power is considered as in VE-MSA and FAST, the average round time can be reduced [see Fig. 7(c)].

## VII. Conclusion

In this article, we investigated a MD selection policy in FL. For this, we formulated a MD selection problem by means of MDP with the objective to select appropriate MDs who can contribute to maximize the increased accuracy per unit-time (i.e., minimize the convergence time). To derive the optimal policy of the formulated problem without any actual FL procedure and prior knowledge on the environment, we introduced a DQN algorithm with the virtual experience replay memory. In this algorithm, to reduce the time of obtaining experiences for the replay memory, the agent virtually executes the selected action and obtains a reward signal (i.e., increased accuracy per unit-time) based on neural networks. Evaluation results demonstrate that the concept of the virtual experience replay memory can significantly reduce the learning time of Q-network by generating lots of experiences within very short time. In addition, it can be found that the derived optimal policy can achieve the shortest convergence time to achieve a target accuracy. In our future work, we will extend our algorithm to balance the numbers of rounds, where different MDs participate by considering the fairness among MDs.

## References

[1] Y. Jing et al., "Crowdtracker: Optimized urban moving object tracking using mobile crowd sensing," *IEEE Internet Things J. (IoT-J)*, vol. 5, no. 5, pp. 3452–3463, Oct. 2018.

[2] D. Kumar et al., "Machine learning algorithms for wireless sensor networks: A survey," *Inf. Fusion*, vol. 49, pp. 1–25, Sep. 2019.

[3] W. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tutorial*, vol. 22, no. 3, pp. 2031–2063, Jul.–Sep. 2020.

[4] H. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[5] K. Toyoda and A. Zhang, "Mechanism design for an incentive-aware blockchain-enabled federated learning platform," in *Proc. IEEE BigData*, 2019, pp. 395–403.

[6] H. Wen, Y. Wu, C. Yang, H. Duan, and S. Yu, "A unified federated learning framework for wireless communications: Towards privacy, efficiency, and security," in *Proc. IEEE INFOCOM Workshops*, 2020, pp. 653–658.

[7] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[8] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–7.

[9] Y. Jin, L. Jiao, Z. Qian, S. Zhang, S. Lu, and X. Wang, "Resource-efficient and convergence-preserving online participant selection in federated learning," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 606–616.

[10] 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 Int. Conf. Commun.*, 2020, pp. 1–7.

[11] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Commun.*, vol. 27, no. 2. pp. 72–80, Apr. 2020.

[12] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[13] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, Nov. 2020.

[14] H. Yang, A. Arafa, T. Q. S. Quek, and H. Vincent Poor, "Age-based scheduling policy for federated learning in mobile edge networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 8743–8747.

[15] N. Yoshda et al., "MAB-based client selection for federated learning with uncertain resources in mobile networks," 2020, *arXiv:2009.13879v1*.

[16] H. Ko, J. Lee, S. Seo, S. Pack, and V. C. M. Leung, "Joint client selection and bandwidth allocation algorithm for federated learning," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3136611.

[17] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1552–1564, Jul. 2021.

[18] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with Non-IID data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24462–24474, 2021.

[19] H. Zhang, Z. Xie, R. Zarei, T. Wu, and K. Chen, "Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 98423–98432, 2021.

[20] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[21] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4385–4395, Mar. 2022.

[22] M. Tang et al., "FedGP: Correlation-based active client selection strategy for heterogeneous federated learning," 2021, *arXiv:2103.13822*.

[23] S. Caldas et al., "LEAF: A benchmark for federated settings," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 1–9.

[24] Wikipedia Cosine Similarity. Accessed: Sep. 17, 2022. [Online]. Available: https://en.Wikipedia.org/wiki/Cosine_similarity

[25] N. Shoham et al., "Overcoming forgetting in federated learning on Non-IID data," 2019, *arXiv:1910.07796*.

[26] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on Non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1698–1707.

[27] W. Shi et al., "Device scheduling with fast convergence for wireless federated learning," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

[28] J. Xu et al., "FedCM: Federated learning with client-level momentum," 2021, *arXiv:2106.10874*.

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

**Mincheol Paik** received the B.S. degree in computer science from the Department of Computer Convergence Software, Korea University, Sejong, South Korea, in 2020. He is currently working toward the M.S. degree with the Department of Computer and Information Science, Korea University, Sejong, South Korea.

His research interests include 5 G networks, network automation, and mobile cloud computing.

**Haneul Ko** (Member, IEEE) received the B.S. and Ph.D. degrees from the School of Electrical Engineering, Korea University, Seoul, South Korea, in 2011 and 2016, respectively, both in electrical engineering.

He is currently an Assistant Professor with the Department of Computer and Information Science, Korea University, Sejong, South Korea. From 2017 to 2018, he was with the Smart Quantum Communication Research Center, Korea University, Seoul, South Korea, and a visiting Postdoctoral Fellow with 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, South Korea. His research interests include 5 G networks, network automation, mobile cloud computing, SDN/NFV, and Future Internet.

**Sangheon Pack** (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer engineering from Seoul National University, Seoul, South Korea, in 2000 and 2005, respectively.

In 2007, he joined the faculty of Korea University, Seoul, Korea, where he is currently a Full Professor with 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. His research interests include future Internet, SDN/ICN/DTN, mobility management, mobile cloud networking, multimedia networking, and vehicular networks.

Mr. Pack was the recipient of KICS (Korean Institute of Communications and Information Sciences) Haedong Young Scholar Award 2013, IEEE ComSoc APB Outstanding Young Researcher Award in 2009, LG Yonam Foundation Overseas Research Professor Program in 2012, and Student Travel Grant Award at the IFIP Personal Wireless Conference (PWC) 2003. From 2002 to 2005, he was a recipient of the Korea Foundation for Advanced Studies Computer Science and Information Technology Scholarship. He was a publication co-chair of IEEE INFOCOM 2014, a co-chair of IEEE VTC 2010-Fall transportation track, a co-chair of IEEE WCSP 2013 wireless networking symposium, a TPC vice-chair of ICOIN 2013, and a publicity co-chair of IEEE SECON 2012. He is an editor of Journal of Communications Networks (JCN).

Authorized licensed use limited to: Kwangwoon Univ. Downloaded on May 30,2023 at 08:35:48 UTC from IEEE Xplore. Restrictions apply.