




Article

Optimal User Selection for High-Performance and Stabilized Energy-Efficient Federated Learning Platforms

Joohyung Jeon ¹, Soohyun Park ², Minseok Choi ³, Joongheon Kim ^{2,*} , Young-Bin Kwon ^{1,*}  and Sungrae Cho ^{1,*} 

¹ School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Korea; joohyung1213@gmail.com

² School of Electrical Engineering, Korea University, Seoul 02841, Korea; soohyun828@korea.ac.kr

³ Department of Telecommunication Engineering, Jeju National University, Jeju 63243, Korea; ejaqmf@jejunu.ac.kr

* Correspondence: joongheon@korea.ac.kr (J.K.); ybkwon@cau.ac.kr (Y.-B.K.); srcho@cau.ac.kr (S.C.)

Received: 21 June 2020; Accepted: 17 August 2020; Published: 21 August 2020



Abstract: Federated learning-enabled edge devices train global models by sharing them while avoiding local data sharing. In federated learning, the sharing of models through communication between several clients and central servers results in various problems such as a high latency and network congestion. Moreover, battery consumption problems caused by local training procedures may impact power-hungry clients. To tackle these issues, federated edge learning (FEEL) applies the network edge technologies of mobile edge computing. In this paper, we propose a novel control algorithm for high-performance and stabilized queue in FEEL system. We consider that the FEEL environment includes the clients transmit data to associated federated edges; these edges then locally update the global model, which is downloaded from the central server via a backhaul. Obtaining greater quantities of local data from the clients facilitates more accurate global model construction; however, this may be harmful in terms of queue stability in the edge, owing to substantial data arrivals from the clients. Therefore, the proposed algorithm varies the number of clients selected for transmission, with the aim of maximizing the time-averaged federated learning accuracy subject to queue stability. Based on this number of clients, the federated edge selects the clients to transmit on the basis of resource status.

Keywords: federated learning; optimization; mobile edge computing

1. Introduction

Deep neural networks have demonstrated strong performances in several machine learning tasks, including speech recognition, object detection, and natural language processing. Using large quantities of training data and complex neural network architectures make it possible to generate high-quality models, which has pushed these systems into applications requiring more computing resources as well as larger and richer datasets. To deal with the larger workloads, data centers have implemented distributed neural network training techniques [1]. To efficiently utilize high-performance computing (HPC) clusters in distributed training, a number of techniques including synchronous and asynchronous updates [2], compression and quantization [3], and hierarchical systems [4] have been considered. However, traditional distributed learning requires the collection and sharing of data from multiple entities to a central data center. Collecting the data from multiple entities to a central data center limits the application of deep learning algorithms to several disciplines which have data privacy-preserving issues. For example, the data created by distributed mobile devices were wealthy

as the data gathered by the central data center, but this rich data is privacy sensitive which may preclude integrating to the central data center. In addition, disciplines such as medicine and finance have employed deep learning algorithms with remarkable success; however, the shortage of data arising from the limitations of sharing privacy sensitive data between data sources is a critical issue [5], as shown in Figure 1.

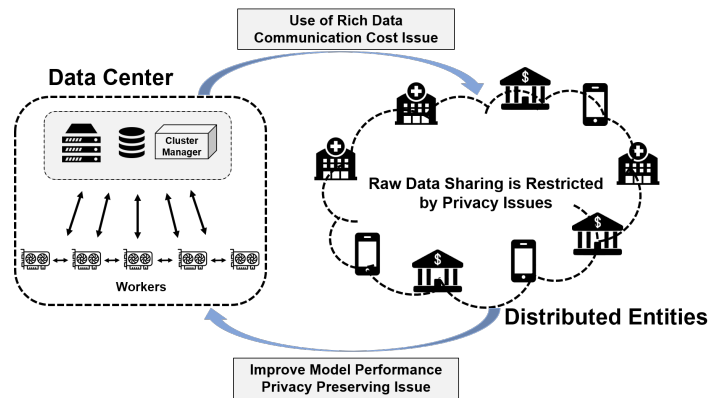


Figure 1. Learning in data centers and distributed domains highlight different aspects of the issue.

In efforts to train models with integrated data, several distributed learning techniques have been investigated. For example, split learning [6,7] is a novel technique for training deep neural networks across multiple data sources; it avoids the sharing of raw data by splitting the sequences of model layers between the client side (data sources) and the server side. However, the utilization of computing resources in split learning is relatively inefficient, due to the sequential training processes of clients [6]. Moreover, communication costs between the clients and the central server increase proportionally to the training dataset size. Although a parallel approach for split learning has been considered [8], communication costs are still dominant in the “cross-device” and “cross-silo” settings, instead of optimizing the distributed processing in the data center [9]. In contrast, federated learning (FL) [10,11] proposes an approach to learn a shared model by aggregating the local updates in a data center while leaving the training data on distributed clients. FL is robust for unbalanced, non-independent and non-identically distributed (non-IID) data, and for systems in which a large number of clients participate [12]. Furthermore, FL can tolerate the participant client drop-outs arising from an unstable environment (i.e., exhaustion of battery, unstable network status etc.) [12]. Several recent studies have applied a mobile edge computing (MEC) structure to FL, resulting in an architecture referred to as federated edge learning (FEEL) [13–15]. FEEL could alleviate the high communication costs by hierarchical architecture.

1.1. Motivation

Although FL trains deep learning model while leaving the data in the client [16], it still exhibits several problems, including high communication costs [17], battery problems of client devices, and unbalanced datasets [18–20]. In particular, the battery consumption incurred by clients as they compute local updates is known to be a constraint of FL procedures; thus, it should be taken into account. In addition, although FL is tolerant toward the dropping out of participating clients, the inclusion of more clients has advantages in terms of training accuracy, owing to the fact that such an inclusion covers larger training datasets. To optimize resource allocation, the authors of [21,22] proposed a client-scheduling algorithm for communication-efficient FL; however, they did not consider the energy consumption problems of power-hungry devices.

In this study, we implement a communication and computation cost adapted federated learning edge architecture and propose the queueing algorithm of federated edge. The federated edge, instead of the clients, performs local updates with data transmitted from clients to reduce the battery consumption and communication burden of the cloud incurred by clients by performing local training. In this structure, the edge acts as a buffer for data uploaded by associated clients; thus, the queueing system in the edge should be taken into consideration. Therefore, we propose an adaptive client number decision algorithm for a stabilized, highly accurate, queue-equipped federated learning edge system using a Lyapunov optimization function. Furthermore, to cover the larger training datasets provided by clients with heterogeneous resources, we propose an energy-aware client-selection method. Our proposed algorithm decides the number of clients and selects the clients. The number of clients is decided considering the queue of the edge and the clients are selected by considering the data amount, communication quality and residual battery power.

The remainder of this paper is organized as follows. Section 2 introduces the background research on FL and an overview of the related work. Section 3 describes the system architecture of our proposed method and Section 4 proposes the proposed adaptive client-selection algorithm. The discussions on security and privacy in FL are briefly introduced in Section 5. In Section 6, we provide a performance evaluation performed by simulations and present a discussion based on the simulation result. In Section 7 we conclude the paper.

2. Federated Learning Edge

FL is a learning technique that can train a deep learning model in a central server with the user data from distributed clients. By computing local updates to the global model from the clients and aggregating the updates in the central server, FL can utilize the computing resources of the distributed clients. FL can train a deep learning model from its distributed users; however, the process incurs large communication costs between the clients and server [23]. Furthermore, performing local updates of the global model is a computationally intensive workload for the client; thus, these updates can entail the consumption of a large proportion of a client's battery. Moreover, the heterogeneity of clients in terms of computational power, data resources, and battery power lead to several issues in FL, such as the straggler problem or wasted resources. Although there are existing studies regarding FL that consider the issues of communication costs [24] and resource allocation [25,26], most have focused on the environment in which the client and central server communicate directly.

The communication burden on the server, generated by the updating of millions of clients, causes significant bottlenecks when scaling up distributed training. To address this communication bottleneck, several techniques, including compression [27,28] and efficient client selection [21,29] have been considered. Various studies have aimed at reducing the communication costs between the clients and server; however, implementing edge computing is the most efficient and practical way to manage numerous clients [13,30–32], as illustrated in Figure 2.

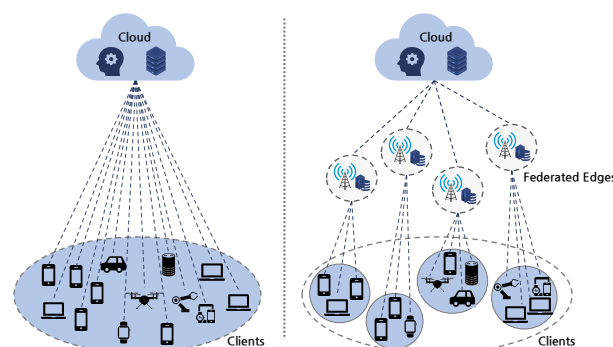


Figure 2. Communication burden comparison for clouds of FL (top) and FEEL (bottom).

Several studies have applied an MCE structure to FL and proposed techniques to optimize the tradeoff between communication and computation overhead. In particular, Wang et al. [33] proposed the cost-efficient method of federated learning across edge nodes. It optimizes communication and computation cost of the edge by load balancing and data scheduling which is leveraged by Lyapunov optimization, however, it uses only a subset of arrived data to train and communication cost exist between the edges while dispatching training data. Xiaofei et al. [34] proposed deep reinforcement learning (DRL) integrated federated learning framework with the mobile edge system. DRL is suitable for optimizing communication and computation resource usage of mobile devices of MEC network, however, the energy consumption issue exists while training DRL agents placed in mobile devices. To the best of our knowledge, this paper is the first work that the edge node which updates the local model selects the optimal user while considering the data amount and battery status to maximize the time-averaged federated learning accuracy subject to queue stability.

3. System Model of Proposed Method

3.1. Clients of the Federated Learning Edge Platform

The system model of this work considers a federated learning edge environment that includes clients, federated edges, and a central cloud server, as illustrated in Figure 3. Clients communicate with the associated federated edge and, if selected by it, send it the collected data. A high-performance computing system and data queue are required in federated edges. Although traditional FL considers the environment in which data-acquiring clients perform local updates of the global model (downloaded from the central server), we consider the environment in which the federated edge gathers data from the associated clients and performs the local updates with this gathered data. Because the federated edge gathers data from the clients and uploads the local updates, it can relieve much of the communication burden of the central server and reduce the long delays incurred through the direct uploading/downloading of local updates/global models by clients. Furthermore, conducting local updates using the copious computation resources of a federated edge could alleviate the battery and straggler problems caused by the consumption of client batteries when performing local updates and the heterogeneity of the clients' computation resources, respectively. Figure 4 shows the reduced battery consumption of client in our system model.

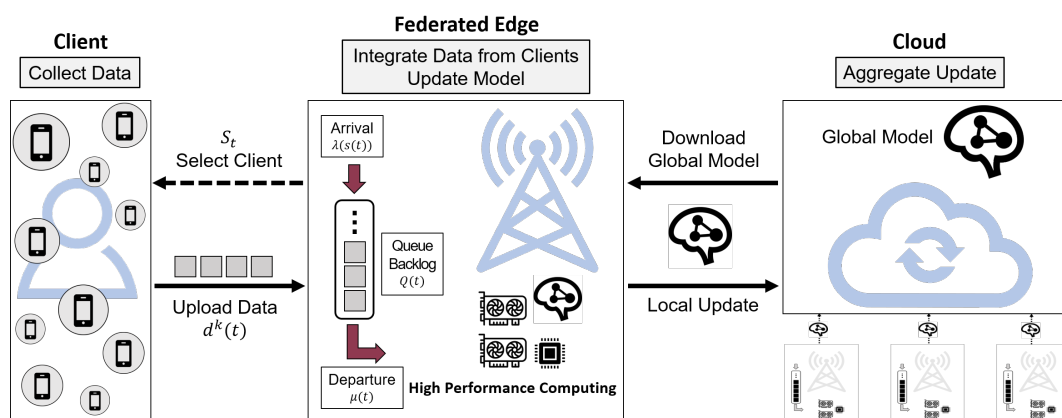


Figure 3. System model of federated learning edge in our proposed algorithm.

The central cloud server broadcasts the global model to the federated edges and aggregates the local updates they provide. Then, the cloud server updates the global model with these aggregated updates. Because the federated edges and the central cloud server are connected via a backhaul link, the federated edges upload the local updates to the cloud server if the communication quality of the backhaul is deemed sufficient for it.

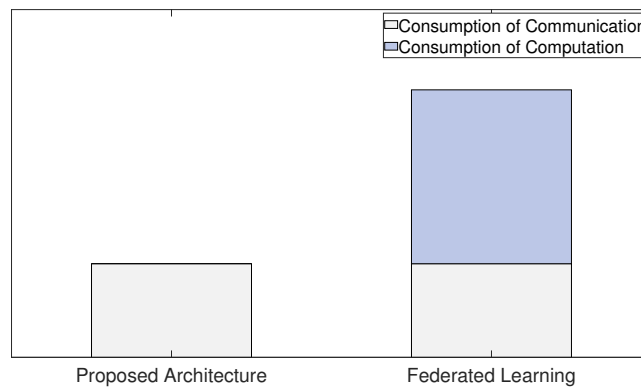


Figure 4. Power consumption of client compare between our system model and standard federated learning. Battery consumption of the client caused by local update computation could be reduced in our system model.

By using federated learning edge architecture to separate the data gathering and local update computation processes between clients and federated edges, our proposed platform can alleviate communication bottlenecks in the central cloud server and reduce the energy demand upon battery-constrained clients.

3.2. Queue-Equipped Federated Edge

In this distributed architecture, the federated edge acts as a buffer, storing the data and computing the local update. Therefore, it is necessary to take into account the data queue existing in the federated edge and the subsequent transmission delays to the cloud server. A departure from the data queue represents the updating of the global model received from the central cloud server, and an arrival in the data queue represents a data transmission from a client, as shown in Figure 5. In the same way as the central server of traditional FL environments selects clients at random to perform the local update, the federated edge in the federated learning edge environment selects the clients which transmit data to it.

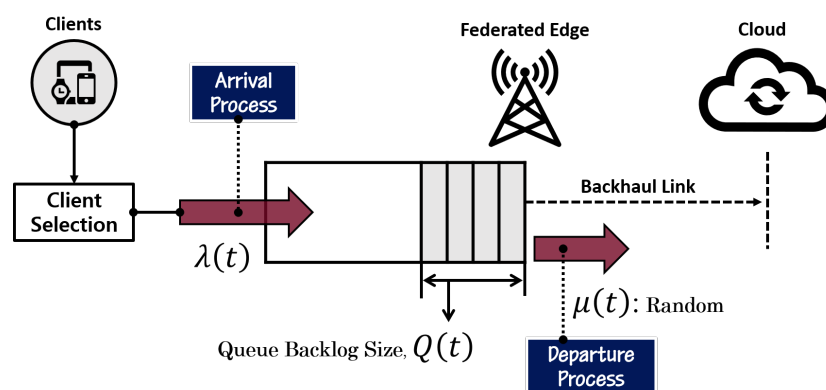


Figure 5. Queue model of data queue in federated edge network.

In the queue-based system, it is necessary to ensure the stability of the queue. The accuracy of the global model increases as greater quantities of data become available for local updates. However, as the queue-backlog cannot extend indefinitely, it is essential to prevent an overflow of the queue. Moreover, as the queue is dependent upon the arrival and departure processes, a stabilized queue can be achieved by ensuring a higher rate of departures than arrivals. Because of the trade-off

between accuracy and queue length, a stochastic, optimization theory-based, time-averaged utility maximization procedure, subject to the system/queue stability algorithm, can be designed.

3.3. Client Selection of Federated Edge

In practical environments, the resources of clients are heterogeneous. Selecting clients at random to upload data to the federated edge can result in the inefficient use of client resources. Therefore, to utilize the heterogeneous resources of clients, our proposed method is designed such that the federated edge selects clients according to their resources, instead of the random selection procedures used by traditional FL methods. In this paper, we take into account the data quantities to be transmitted as well as the residual battery power and communication quality of the data-acquiring clients. After the federated edge has decided upon the number of clients to receive the data from, it selects the clients on the basis of priorities determined by the resources of each client.

4. Proposed Algorithm

In this section, the details of our proposed client selection algorithm are introduced. Our proposed client selection algorithm includes two steps: deciding upon the number of clients and selecting as many clients as previously specified. The federated edge decides the number of clients to receive data from using the control-Lyapunov function for a stabilized data queue in federated learning edge. Next, the edge selects the clients according to a weighting that is calculated from the resource status of each client. Because we assume that the clients selected to send the data each transmit an equal amount of data to the federated edge, the number of clients can be decided using the data queue length of the federated edge. The notations for the parameters used in this paper, along with their descriptions, are summarized in Table 1. The procedures at the federated edge and each clients are presented in Algorithms 1 and 2, respectively. In each unit time t , the federated edge proceeds two steps which are deciding the optimal number of clients and selecting clients. Each client k proceeds function **SendStatus_k(t)** and **SendData_k(t)** which sends the data amount, communication quality, and residual battery power to the federated edge and sends the training data to the federated edge if the functions are called, respectively.

Table 1. Notations.

Parameter	Description
K	Total number of clients
n^k	Number of data of client k
$Comm^k$	Communication quality of client k
B^k	Residual battery power of client k
p^k	Weight of client k
s^*	Time-averaged optimal client number
$s(t)$	Possible client numbers at t
\mathcal{X}	Set of client numbers
V	Trade-off factor between accuracy and queue-backlog
$U(s(t))$	Utility function when client number $s(t)$ is given
$Q(t)$	Federated edge queue size at t
$\lambda(s(t))$	Arrival process when client number $s(t)$ is given
$\mu(t)$	Departure process at t
f	Carrier frequency
T_{power}	Transmission power
$L_k(t)$	Path loss at t
$dist_k(t)$	Distance between client k and federated edge at time t

Algorithm 1 Procedure at the federated edge.

```

1:  $K$  : Total number of clients
2:  $Q(t)$  : Federated edge queue size at  $t$ 
3:  $\mathcal{X}$  : The set of client numbers
4:  $V$  : Trade-off factor between accuracy and queue-backlog
5:  $S_t$  : The set of selected clients at  $t$ 
6:  $P_t$  : The array of priorities of clients at  $t$ 
7:  $P_t[i]$  :  $i$ -th element of array  $P_t$  at  $t$ 
8:  $s^*(t)$  : Time-average optimal client number at  $t$ 
9:  $d^k(t)$  : Data transmitted from client  $k$  at  $t$ 
10:  $r^k$  : Timeout value for receiving resource status from client  $k$ 
11:  $Q[t] \leftarrow 0$ 
12: for each unit time  $t = 0, 1, 2, \dots$  do
13:   Step1 : Optimal number of clients decision
14:   Observe  $Q[t]$ 
15:    $\mathcal{T}^* \leftarrow -\infty$ 
16:   for  $s(t) \in \mathcal{X}$  do
17:      $\mathcal{T} \leftarrow V \cdot U(s(t)) - Q(t) \cdot \{\lambda(s(t)) - \mu(t)\}$ 
18:     if  $\mathcal{T} \geq \mathcal{T}^*$  then
19:        $\mathcal{T}^* \leftarrow \mathcal{T}$ 
20:        $s^*(t) \leftarrow s(t)$  // Optimal number of clients
21:     end if
22:   end for
23:   Step2 : Client selection
24:   Initialize  $S_t = \emptyset$  and  $P_t = [p^1(t), p^2(t), \dots, p^K(t)]$ , where  $p^i(t), \forall i \in \{1, \dots, K\}$ 
25:   for each client  $k = 1, 2, \dots, K$  in parallel do
26:     Call  $\text{SendStatus}_k(t)$  of client  $k$ 
27:     Start timer( $r^k$ )
28:     Wait until(receipt of reply from client  $k$  OR timeout)
29:     if receipt of reply from client  $k$  then
30:        $n^k(t), \text{Comm}^k(t), B^k(t) \leftarrow \text{SendStatus}_k(t)$ 
31:       if  $B^k(t) = 0$  then // If residual battery power is 0
32:          $p^k(t) \leftarrow 0$ 
33:       else
34:          $p^k(t) \leftarrow \frac{n^k(t) \cdot \text{Comm}^k(t)}{B^k(t)}$  // Priority value of client  $p^k(t)$ 
35:       end if
36:     else // No reply from client  $k$  until timeout
37:        $p^k(t) \leftarrow 0$ 
38:     end if
39:     Reset timer( $r^k$ )
40:   end for
41:   Sort  $P_t$  in descending order
42:   for each element  $P_t[i] \in P_t, 0 \leq i \leq s^*(t)$  do // For selected clients
43:      $S_t = S_t \cup P_t[i]$  //  $P_t[i]$  is  $i$ -th element of sorted array  $P_t$ 
44:   end for
45:   for each client  $k \in S_t$  in parallel do
46:      $d^k(t) \leftarrow \text{SendData}_k(t)$ 
47:   end for
48:    $Q(t+1) \leftarrow \max\{Q(t) - \mu(t), 0\} + \sum_{k \in S_t} |d^k(t)|$ 
49: end for

```

Algorithm 2 Procedure at each client k .

```

1: SendStatus $_k(t)$ :
2:    $n^k(t) \leftarrow$  (Data amount of client  $k$  at  $t$ )
3:    $Comm^k(t) \leftarrow$  (Communication quality of client  $k$  at  $t$ )
4:    $B^k(t) \leftarrow$  (Residual battery power of client  $k$  at  $t$ )
5:   return  $(n^k(t), Comm^k(t), B^k(t))$  to federated edge
6: SendData $_k(t)$ :
7:    $d^k(t) \leftarrow$  (Defined amount of data to send)
8:   return  $d^k(t)$  to federated edge

```

4.1. Client Number Control by Lyapunov Optimization

Then, the queue at the edge can be formulated as follows:

$$Q(t+1) = \max\{Q(t) - \mu(t), 0\} + \lambda(t) \quad (1)$$

where $Q(t)$, $\lambda(t)$, and $\mu(t)$ indicate the queue-backlog size at the edge at time t , the amount of data arriving from the associated clients at time t , and the amount of data departing to the cloud server at time t , respectively. In this case, the departure at time t (that is, $\mu(t)$) is not controllable because the edge can only transmit data from the queue to the extent that the wireless channel (between the edge and the cloud server) permits it. The quantity of data arriving from the clients to the edge is controllable, being dependent on the number of selected clients.

This section formulates the time-averaged FL accuracy maximization, which is subject to queue stability constraints; here our control action is the number of clients. The mathematical program for this time-averaged optimization can be formulated as follows:

$$\max : \quad \lim_{t \rightarrow \infty} \sum_{\tau=0}^{t-1} U(s(\tau)) \quad (2)$$

$$\text{s.t.} \quad \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Q(\tau) < \infty \quad (3)$$

where (3) represents the queue stability constraint. In (2), we can observe that our utility function when client number $s(t)$ is given is denoted as $U(s(t))$. The utility function can be expressed as follows $U(s(t)) \triangleq A(s(t))$ where $A(s(t))$ stands for the expected accuracy of the learning model when client number $s(t)$ is given.

According to what Neely said in his book [35], the general form of the optimization equation can be expressed as follows:

$$x^*(t) \leftarrow \arg \max_{x(t) \in \mathcal{X}} [V \cdot U(x(t)) - Q(t) \cdot \{\lambda(x(t)) - \mu(x(t))\}] \quad (4)$$

where $x^*(t)$, $x(t)$, \mathcal{X} , V , $U(x(t))$, $Q(t)$, $\lambda(x(t))$, and $\mu(x(t))$ stand for time-average optimal solution, possible solution, set of possible solutions, accuracy-delay tradeoff factor, utility function when solution x is given, queue backlog, arrival process when solution x is given, and departure process when x is given, respectively.

According to the Lyapunov optimization framework, and by implementing the so-called drift-plus-penalty (DPP), this time-averaged optimization framework can be revised under the constant-gap approximation as follows [36]:

$$s^*(t) \leftarrow \arg \max_{s(t) \in \mathcal{X}} [V \cdot U(s(t)) - Q(t) \cdot \{\lambda(s(t)) - \mu(t)\}] \quad (5)$$

where $s^*(t)$, $s(t)$, \mathcal{X} , V , $U(s(t))$, $Q(t)$, $\lambda(s(t))$, and $\mu(t)$ represent the time-averaged optimal client number, the possible number of clients, the set of client numbers, the trade-off factor between

accuracy and queue length, the utility function when client number $s(t)$ is given, the queue length, the arrival process when client number $s(t)$ is given, and the departure process, respectively.

In the aforementioned system model, the federated edge should receive data from data-acquiring clients after considering the data queue in the edge. Although the accuracy of the learning model increases when more data is used, achieving queue stability should take a higher priority. Under this condition, we can design a dynamic algorithm that selects the number of clients from which to receive data for each unit time, in a way that maximizes the time-averaged accuracy (subject to queue stability), as we did in (5).

In every unit time, the clients which are available to send datasets notify the federated edge of their quantity of data, communication quality, and residual battery power. The condition of each client is determined internally according to its task priority. Then, the federated edge selects the number of clients to receive the data from, using a Lyapunov control-based time-averaged optimization function.

If the data queue of the federated edge contains a substantial amount of data, the optimization function selects fewer clients to transmit. Furthermore, if the data queue is almost empty, the optimization function selects numerous clients, thereby improving the accuracy of the model using larger datasets. Deciding the optimal client number is described in Line 13–22 of Algorithm 1. After observing queue-backlog in Line 14, through comparing the value of optimization function about all $s(t) \in \mathcal{X}$ (Line 15–22), the federated edge decides optimal client number $s^*(t)$ which maximizes the time-average accuracy and also stabilizes the queue.

4.2. Client Selection

After the number of clients to receive the data from has been decided upon by the optimization function, the federated edge selects the clients with which to upload the data to the edge. Traditional FL selects a random client for the update; however, this approach can lead to an inefficient use of the network and heterogeneous resources of the mobile clients. Therefore, our algorithm selects the clients on the basis of their data queue size, communication quality, and residual battery power; furthermore, it operates efficiently by considering the heterogeneous resources of the clients. Because the available clients send the information concerning their resources, the federated edge can select the clients according to their resource statuses without additional information exchange. The client selection flow is illustrated in Figure 6.

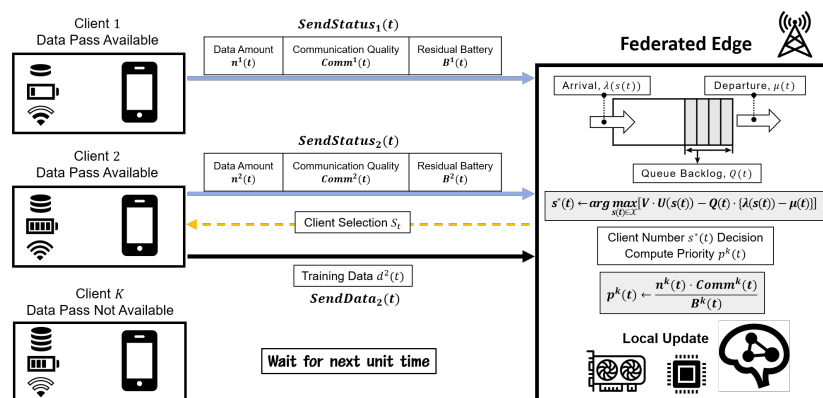


Figure 6. The federated edge selects the number of clients to receive the data from and selects these clients from amongst those available, according to each client's resource status.

The weight of client k , which is the criteria for selecting the clients, can be formulated based on their resource status, as follows:

$$p^k = \frac{n^k \cdot \text{Comm}^k}{B^k} \quad (6)$$

where n^k , $Comm^k$, and B^k represent the data amount of the client, its communication quality, and residual battery k , respectively. The weighting of each client increases when they have more data, better communication quality, or lower battery power. The battery power is inversely proportional to the weight because it is more effective to send the data before the client runs out of battery, thus we increase the accuracy of the learning model. The data amount is defined as the size of the data which could be represented by a byte. Communication quality is defined as channel state information. The residual battery could be represented by Wh.

The procedure of selecting clients is described in Line 23–49 of Algorithm 1. In Line 24 the federated edge initializes S_t the set of selected clients at t to empty set and P_t the array of priorities of clients at t with each element $p^1(t), p^2(t), \dots, p^k(t)$. Then the federated edge calls the **SendStatus_k(t)** function of all clients in Line 26 and starts the timer r^k of each client k to check the timeout of clients in Line 27. If the federated edge receives the reply from client k before the timeout, the federated edge could get $n^k(t), Comm^k(t), B^k(t)$ which are data amount, communication quality, and residual battery of client k , respectively, returned by **SendStatus_k(t)** (Line 30). Although the federated edge receives the resource status of client k , returned residual battery power $B^k(t)$ could be 0, then $p^k(t)$ the weight of client k be 0 (Line 31 and 32). On the other hand, $p^k(t)$ the weight of client k is computed as $p^k(t) \leftarrow \frac{n^k(t) \cdot Comm^k(t)}{B^k(t)}$ (Line 34). If the federated edge didn't receive a reply from client k before the timeout, the weight $p^k(t)$ of client k is 0 (Line 37).

After the weight of each client is decided, the federated edge sorts P_t the array of priorities in descending order (Line 41). Then S_t the set of selected clients is composed of $s^*(t)$ clients with the largest weight in array P_t (Line 42–44). As the clients which to send the data to the federated edge are selected, the federated edge calls **SendData_k(t)** function for each client k in set S_t and returned data is $d^k(t)$ (Line 45–47). Then the federated edge updates the queue-backlog by $\mu(t)$ the amount of data departure at t and $\sum_{k \in S_t} d^k(t)$ the amount of data arrival at t which is $\lambda(s(t))$ of (4.4) (Line 48).

Client k executes two functions **SendStatus_k(t)** and **SendData_k(t)** if they are called by the federated edge as shown in Algorithm 2. Function **SendStatus_k(t)** returns $n^k(t), Comm^k(t)$, and $B^k(t)$, which are the data amount of client k at t , communication quality of client k at t , and residual battery power of client k at t , respectively (Line 1–5). If the function **SendData_k(t)** is called by the federated edge, it returns $d^k(t)$ the defined amount of data to send (Line 6–8).

After the clients have been prioritized according to their weights p^k , the top $s^*(t)$ clients send the data to the federated edge. The federated edge continues the FL process with the data in its data queue and the local model downloaded from the central cloud server. The entire process of our proposed federated learning edge system, with adaptive client selection for stabilized and highly accurate edge platform, is described in Algorithms 1 and 2.

In our proposed algorithm, which is inspired by the Lyapunov optimization, shows low time complexity due to the fact that it (i) computes (5) by applying the given set of candidates and then (ii) find the argument which shows the minimum of the result of (5). Thus, its run-time computational complexity is $O(N)$.

5. Security and Privacy Discussions in FL

In FL, the security and privacy issues are de facto problems to be addressed. The reason why we are using FL is that we do not want to gather all data in a centralized single storage. It means that we want to maintain certain amounts of security and privacy. Thus, the related discussions are essential.

In our proposed system, we select certain amounts of clients for conducting local deep neural network training in each edge. For the operation, our proposed algorithm selects the clients in terms of pre-determined priority. If we can newly define the priority in terms of user-privacy or security-levels, we can re-build the FL system under the consideration of security and privacy. Therefore, there are some chances in order to consider security levels.

Preserving data privacy should be achieved in the scope of user or organization, not the data acquiring device. Therefore, if a user with multiple devices which acquire data or an organization with multiple data center could define privacy and security-levels in bounds of user or organization, not the device. For example, a user who owns several IoT devices such as smartphones, health care devices, smartwatches could conduct local updates using smart speaker which receives a steady supply of power. A smart speaker receives the data from each IoT devices and conducts the local update as the federated edge. Then upload the update to the cloud. Although the data of acquiring device are transmitted to the edge the data privacy of the user is preserved.

Essentially, if we conduct local model training, and then aggregate the local training parameters in order to build our desired global model, it already achieves privacy preserving because we did not reveal our own data themselves. On top of this impact about privacy-preserving, if we can define privacy-aware client selection, then we can consider more privacy and security concerns.

6. Performance Evaluation

In this section, we evaluate our proposed adaptive client-selection method for stabilized and highly accurate federated learning edge platforms. We simulate the queue stability of federated edge which the expected total accuracy of the model is based on the learning curve [37].

6.1. Experiment Setting

Our simulation is designed and implemented based on the environment described in Section 3. To evaluate our proposed algorithm, which stabilizes the data queue and achieves a high utility function, we assumed that the queue of a federated edge and 50 associated clients are used to transmit the data in the simulation.

The federated edge trains the global model received from the central cloud and the training data is fetched from the data queue of the federated edge. The collection of data, which represents a departure from the data queue, is decided upon by the FL protocol between the federated edge and central cloud. Although several parameters that control the local update procedure are controlled by the central cloud, the communication between the federated edge and central cloud is random. Therefore, we assumed that departures from the data queue of the federated edge are independent and identically random. An arrival to the data queue is determined from the quantities of data of selected clients.

We assumed 1 federated edge and total client number $k = 50$ in the simulation. Initially, the battery level of each client is set at random and the residual battery of every client decreases for each unit time. Wireless communications between federated edge and clients were modeled based on LTE networks. The federated edge is located at the center of the cell with a radius of 100 m, and 50 clients are distributed in the cell. The carrier frequency f , transmission power T_{power} , and bandwidth is 2.5 GHz, 20 dBm, and 20 MHz, respectively. The communication quality $Comm^k(t)$ between the federated edge and client k at time t is calculated as follows:

$$L_k(t) = 20 \log_{10}(f) + 10N \log_{10} \{dist_k(t)\} - 28 \quad (7)$$

$$Comm^k(t) = \{T_{power} - L_k(t)\} \times Noise \quad (8)$$

where $L_k(t)$ and $dist_k(t)$ represents the path loss [38] and distance between the federated edge and the client k at time t . The unit of path loss is dBm and follows a power law with the exponent $N = 3$. The distance $dist_k(t)$ varies randomly at every unit time from 1 m to 100 m. The noise and fading could affect the received power. Therefore $Noise$ value from 0 to 1 multiplied in (8). The communication quality is not good if the $Comm^k(t)$ value is closer to 0, and the closer to 1, the better.

We assumed 100,000 training data, which are partitioned between 50 clients, each receiving 2000 data. The selected clients transmit 10 samples to the federated edge per unit time. Each data is an image of 3 MB size. Data size is decided by the image size distribution of the author's mobile device.

The model of the mobile device is Galaxy S8 which equipped 12M pixel camera and 3000 mAh battery. Figure 7 shows the image size distribution of 1700 data. The average size of data is 3 MB and decided as the data size for the simulation. The initial values of V and Q_{max} are $V = 10^9$ and $Q_{max} = 2 \times 10^3$. In practice, Q_{max} can be determined depending on the system, then V can be properly selected. The utilization function $U(s(t))$ is assumed as the anticipated accuracy when the number of clients is decided as $s(t)$. This expected accuracy is modeled by the learning curve [37], as shown in Figure 8.

In order to evaluate our proposed algorithm, we compared it against three client-selecting methods:

1. Max Selection: The federated edge receives data from every client at every unit time.
2. Static Selection: The federated edge selects the same amount of clients at every unit time. In this evaluation, five clients were used to transmit the data for each unit time.
3. Random Selection: The number of selected clients is decided in the same way as our proposed algorithm; however, it selects random clients without considering the resources of the clients.

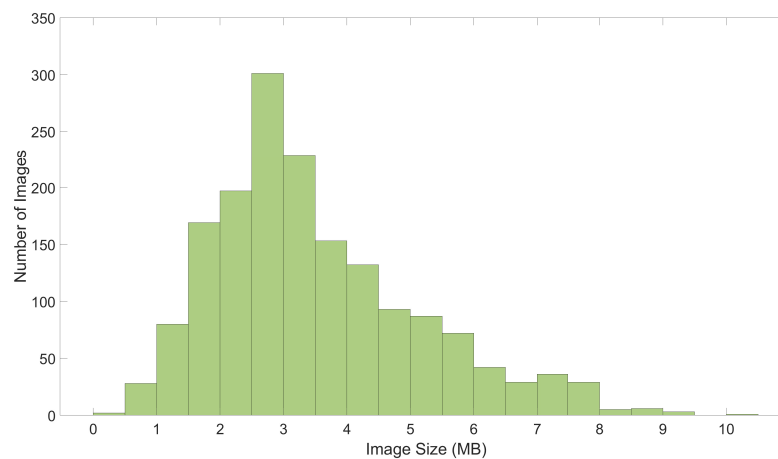


Figure 7. Data size distribution of author's mobile device.

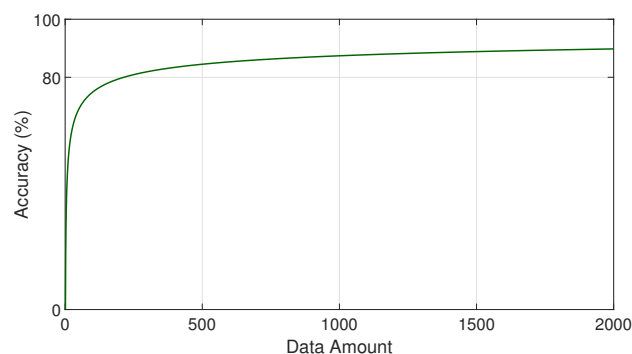


Figure 8. Expected accuracy under increasing amount of dataset.

As a benchmark scheme to compare against our algorithm, we consider a FL procedure that selects random clients from a static number of clients, where the client fraction $C = 0.1$ [10]. The number of clients for each unit time of the first two methods are constant, thus we can compare the queue stability and achieve a long queue backlog, which is related to the utility function under our proposed algorithm. In addition, the last case is considered to evaluate the resource-aware client selection of our proposed algorithm. It decides the number of clients in a similar way to

our proposed algorithm, but selects random clients without considering their available resources. Every method selects the client randomly after deciding upon the number of clients to receive the data from; however, the last one decides the number of clients in a manner identical to that of our proposed algorithm. Our proposed algorithm decides the number of selected clients for each unit time according to the queue status, and selects the clients according to their resource statuses.

6.2. Experimental Results

To evaluate our proposed algorithm, we chart the queue backlog plots against time t in Figure 9. When max selection and static selection are performed to decide the number of selected client, the federated edge system is unstable due to queue overflow and achieves an extremely stable queue, respectively. However, as the main object of our federated learning edge system is to maximize the learning accuracy, our proposed algorithm and random selection, which are controlled by the same client number decision, both select the same optimal number of clients to stabilize the queue. After $t = 700$, the queue backlog of random selection slightly decreases and after $t = 750$ the queue backlog of our proposed algorithm also decreases. As time passes, depletion of battery and a lack of data to transmit from clients leads to a decrease in the available number of clients, this is less than the client number required to maximize utilization. Because the random selection selects the client without considering the resources of each client, some clients with sufficient data but insufficient battery or vice versa become unavailable at the later time slots. On the other hand, our proposed algorithm considers the resources available (in particular the battery level of the clients) and thereby achieves a larger utilization by receiving more data from the entire set of clients.

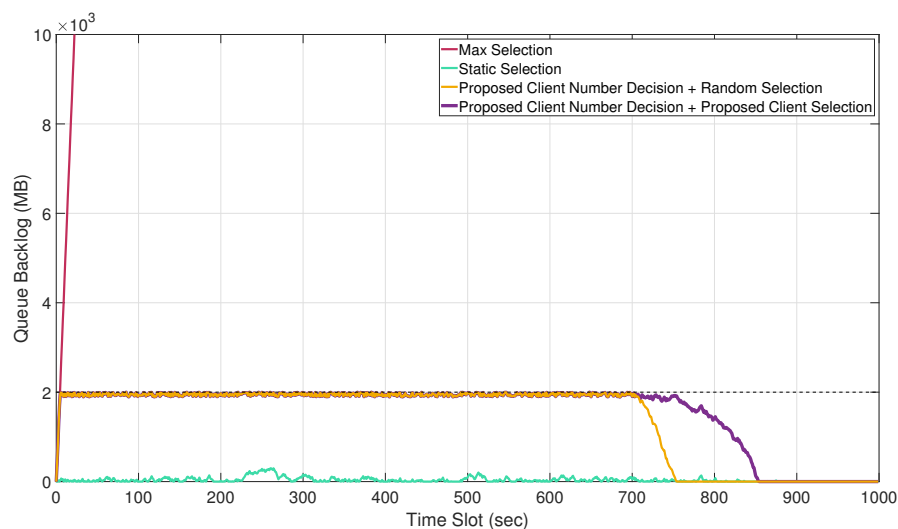


Figure 9. Queue backlog variation plots (y) versus time slot (x). The decided number of clients of the proposed algorithm and random selection is same. However, proposed algorithm selects the clients which to transmit the data by our proposed client selection which considers the resource and random selection just randomly selects the clients.

Figure 10a shows the total number of communications of all clients. Our proposed algorithm communicates 4133 times and random selection communicates 3621 times. As the total quantity of uploaded data is proportional to the number of communications, our proposed algorithm uploaded more data than random client selection. As a result, our proposed algorithm could receive more data from the clients which results to training more accurate model. Figure 10b shows the compare of expected learning accuracy of proposed client selection and random client selection based on learning curve and total number of received data. Proposed algorithm and random selection achieve 88% and 80%, respectively.

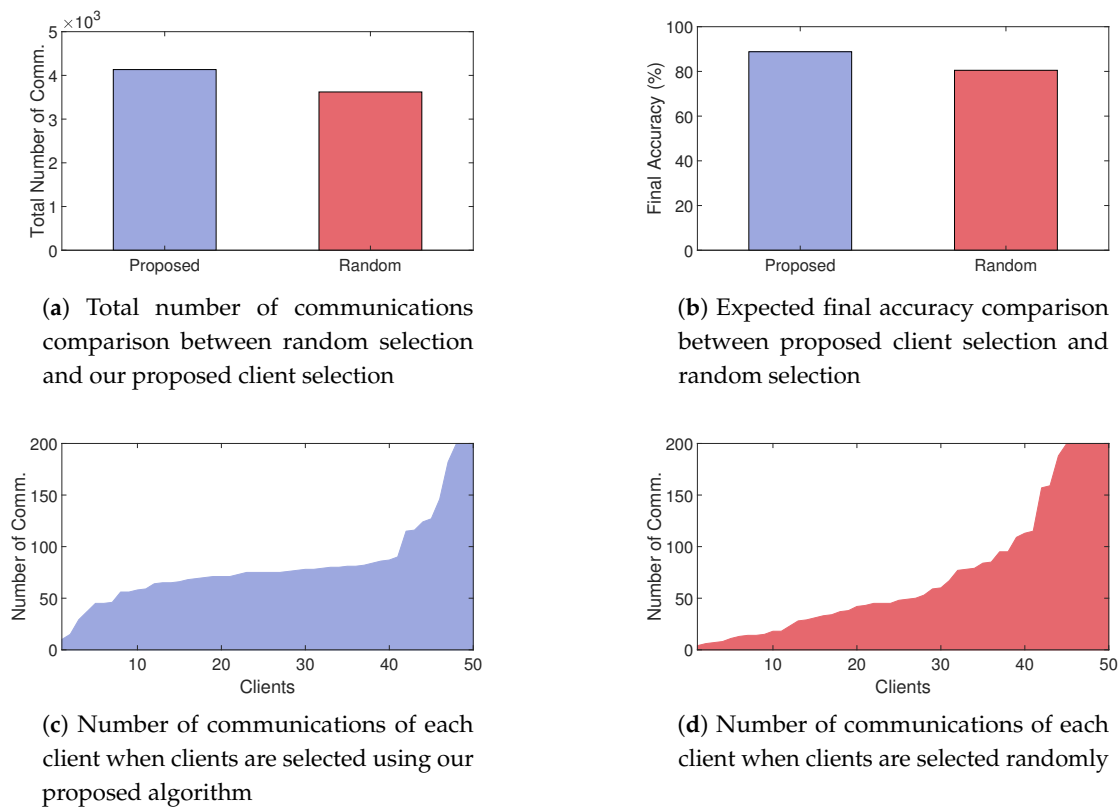


Figure 10. (a) Total number of communications between the federated edge and associated clients; (b) represents the expected final accuracy based on the learning curve and total number of received data; (c,d) represent the number of communications for each client when clients are selected using our proposed algorithm and at random, respectively. (c,d) are sorted by number of communications. The variance in the number of communications for each client under our proposed algorithm and random client selection is 41.76 and 62.7, respectively.

Our proposed algorithm could also improve the fairness of client selection. Improved fairness can decrease the likelihood of overfitting, which can arise from unbalanced training data exchange between clients. Figure 10c,d show the number of communications of each client when the clients are selected using our proposed algorithm and at random, respectively. Although random selection shows large differences in the number of communications between clients, our proposed client selection shows relatively fair communication numbers between clients. Comparing the two client-selection algorithms, we see that the variance value of the proposed algorithm is 41.76, this is reduced from the variance under random selection, which is 62.7. Figure 11 shows the histogram of the number of clients for each number of communications. The communication counts for each client of random selection are unbalanced. Because the communication counts differ widely between clients, some of the clients send large amounts of data and others send less. The top 10 clients of random selection, which comprise 20% of the total set of clients, communication 47% of the total communications. These clients are the clients which have more battery and more data but didn't selected at the early time slots and selected frequently at the late time slots.

To conclude, our proposed algorithm improves not only the utilization of client resources while stabilizing the queue but also improves the fairness of client selection.

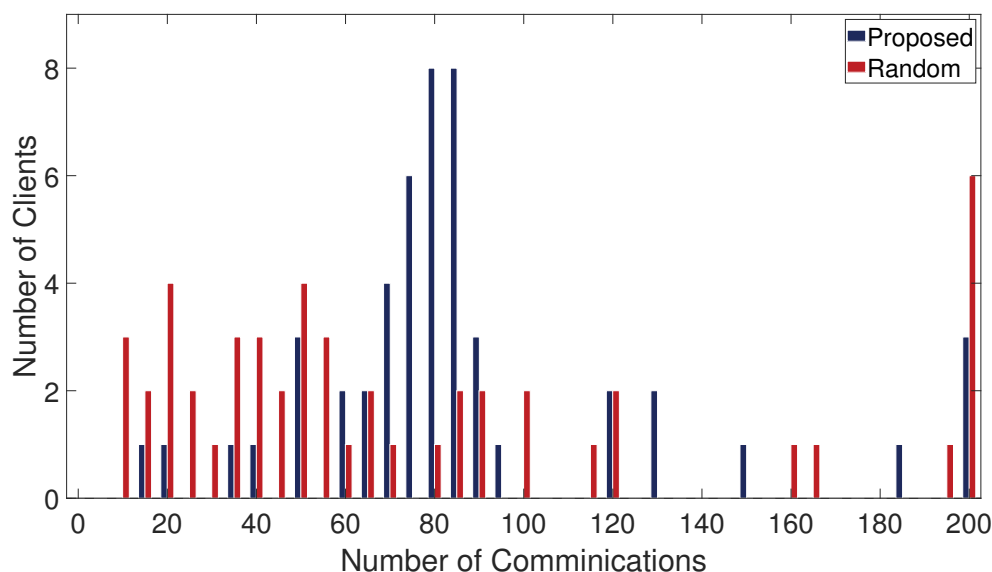


Figure 11. Number of clients for each number of communications.

7. Concluding Remarks and Future Work

In this paper, we propose an adaptive client-selection algorithm that stabilizes the data queue of a federated edge while also realizing a highly accurate learning model in the energy-aware federated learning edge environment. By conducting the local computation in the federated edge instead of the power-consuming clients, clients instead consume a relatively small amount of battery power, only being required to transmit a dataset. In our proposed algorithm, the federated edge adaptively decides upon the number of clients to receive the data from, according to Lyapunov control-based time-averaged optimization function. This client selection algorithm maximizes the time-averaged accuracy (subject to stability), balancing the trade-off between accuracy and stability. Moreover, the federated edge selects the clients to send the data to according to the priorities of the clients, this is decided on the basis of their quantity of data, communication quality, and residual battery power. The evaluation results show that our proposed algorithm can maximize the time-averaged utilization subject to stability, balancing the trade-off between stability and accuracy. Specifically, by considering the battery power of clients, our algorithm makes use of the heterogeneous resources of clients and achieves a significant increase in fairness compared to random client-selection procedures. In addition, our proposed algorithm transmits more data than random client selection which leads to more accurate model and the reduced variance in the communication counts of clients demonstrates the increased fairness between the clients.

In future work, the federated edge could consider the scheduling of multiple data queues for multiple learning tasks. Furthermore, to employ the data of UAVs or vehicles, the federated edge could consider communication and computing resource allocation for clients with high mobility.

Author Contributions: J.J., M.C., J.K., and S.C. were the main researchers who initiated and organized the research reported in the paper, and all authors including S.P. and Y.-B.K. were responsible for analyzing the simulation results and writing the paper. S.P. designed a more realistic network scenario and a subsequent suitable algorithm modification. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Chung-Ang University Graduate Research Scholarship in 2018 (for Joohyung Jeon), by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) under Grant 2017-0-00068, A Development of Driving Decision Engine for Autonomous Driving using Driving Experience Information, and also by Chung-Ang University Research Grant in 2017 for Young-Bin Kwon (1 September 2017–28 February 2018).

Acknowledgments: J.K., Y.-B.K. and S.C. are the corresponding authors of this paper.

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

References

1. Sze, V.; Chen, Y.H.; Yang, T.J.; Emer J.E. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* **2017**, *105*, 2295–2329. [[CrossRef](#)]
2. Zhang, W.; Gupta, S.; Lian, X.; Liu, J. Staleness-Aware Async-SGD for Distributed Deep Learning. *arXiv* **2015**, arXiv:1511.05950.
3. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2015**, arXiv:1510.00149.
4. Gupta, S.; Zhang, W.; Wang, F. Model Accuracy and Runtime Tradeoff in Distributed Deep Learning: A Systematic Study. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016.
5. Jeon, J.; Kim D.; Kim J. Cyclic Parameter Sharing for Privacy-Preserving Distributed Deep Learning Platforms. In Proceedings of the International Conference on Artificial Intelligence in Information and Communication (ICAIIIC), Okinawa, Japan, 11–13 February 2019.
6. Gupta, O.; Raskar, R. Distributed Learning of Deep Neural Network over Multiple Agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [[CrossRef](#)]
7. Jeon, J.; Kim, J.; Kim, J.; Kim, K.; Mohaisen, A.; Kim, J. Privacy-Preserving Deep Learning Computation for Geo-Distributed Medical Big-Data Platforms. In Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) Supplemental Volume, Portland, OR, USA, 24–27 June 2019.
8. Jeon, J.; Kim, J. Privacy-Sensitive Parallel Split Learning. In Proceedings of the IEEE International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 January 2020.
9. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. *arXiv* **2019**, arXiv:1912.04977.
10. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 20–22 April 2017.
11. Konečný, J.; McMahan, H.B.; Ramage, D. Federated Optimization: Distributed Optimization Beyond the Datacenter. In Proceedings of the NIPS Workshop on Optimization for Machine Learning, Montreal, QC, Canada, 11 December 2015.
12. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [[CrossRef](#)]
13. Wang, S.; Tuor, T.; Salonidis, T.; Leung, K.K.; Makaya, C.; He, T.; Chan, K. When Edge Meets Learning: Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1205–1221. [[CrossRef](#)]
14. Zhu, G.; Wang, Y.; Huang, K. Broadband Analog Aggregation for Low-Latency Federated Edge Learning. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 491–506. [[CrossRef](#)]
15. Amiri, M.M.; Gündüz, D. Machine Learning at the Wireless Edge: Distributed Stochastic Gradient Descent Over-the-Air. In Proceedings of the IEEE International Symposium on Information Theory (ISIT), Paris, France, 7–12 July 2019.
16. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), Dallas, TX, USA, 30 October–3 November 2017.
17. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, H.B.; et al. Towards Federated Learning at Scale: System Design. In Proceedings of the Conference on Systems and Machine Learning (SysML), Palo Alto, CA, USA, 31 March–2 April 2019.
18. Sattler, F.; Wiedemann, S.; Müller, K.R.; Samek, W. Robust and Communication-Efficient Federated Learning from Non-IID Data. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**. [[CrossRef](#)]
19. Smith, V.; Chiang, C.K.; Sanjabi, M.; Talwalkar, A.S. Federated Multi-Task Learning. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), Longbeach, CA, USA, 4–9 December 2017.

20. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated Learning with Non-IID Data. *arXiv* **2018**, arXiv:1806.005829.
21. Nishio, T.; Yonetani, R. Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In Proceedings of the IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019.
22. Wadu, M.M.; Samarakoon, S.; Bennis, M. Federated Learning under Channel Uncertainty: Joint Client Scheduling and Resource Allocation. *arXiv* **2020**, arXiv:2020.00802.
23. Tran, N.H.; Bao, W.; Zomaya, A.; Nguyen, M.N.H.; Hong, C.S. Federated Learning over Wireless Networks: Optimization Model Design and Analysis. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Paris, France, 29 April–2 May 2019.
24. Jeong, E.; Oh, S.; Kim, H.; Park, J.; Bennis, M.; Kim, S. Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data. *arXiv* **2018**, arXiv:1811.11479.
25. Samarakoon, S.; Bennis, M.; Saad, W.; Debbah, M. Distributed Federated Learning for Ultra-Reliable Low-Latency Vehicular Communications. *IEEE Trans. Commun.* **2019**, *68*, 1146–1159. [[CrossRef](#)]
26. Samarakoon, S.; Bennis, M.; Saad, W.; Debbah, M. Federated Learning for Ultra-Reliable Low-Latency V2V Communications. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, UAE, 9–13 December 2018.
27. Konečný, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2016**, arXiv:1610.05492.
28. Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W.J. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
29. Yang, H.H.; Arafa, A.; Quek, T.Q.S.; Poor, H.V. Age-Based Scheduling Policy for Federated Learning in Mobile Edge Networks. *arXiv* **2019**, arXiv:1910.14648.
30. Abad, M.S.H.; Ozfatura E.; Gunduz D.; Ercetin, O. Hierarchical Federated Learning Across Heterogeneous Cellular Networks. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020.
31. Park, J.; Samarakoon, S.; Bennis, M.; Debbah, M. Wireless Network Intelligence at the Edge. *Proc. IEEE* **2019**, *107*, 2204–2239. [[CrossRef](#)]
32. Yang, Q.; Liu, Y.; Cheng, Y.; Kang, Y.; Chen, T.; Yu, H. *Horizontal Federated Learning, Federated Learning Synthesis Lectures on Artificial Intelligence and Machine Learning*; Morgan & Claypool: San Rafael, CA, USA, 2019; pp. 63–64.
33. Wang, X.; Han, Y.; Wang, C.; Zhao, Q.; Chen, X.; Chen, M. In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning. *IEEE Netw.* **2019**, *33*, 156–165. [[CrossRef](#)]
34. Zhou, Z.; Yang, S.; Pu, L.; Yu, S. CEFL: Online Admission Control, Data Scheduling and Accuracy Tuning for Cost-Efficient Federated Learning Across Edge Nodes. *IEEE Internet Things J.* **2020**. [[CrossRef](#)]
35. Neely, M. *Stochastic Network Optimization with Application to Communication and Queueing Systems*; Morgan & Claypool: San Rafael, CA, USA, 2010.
36. Kim, J.; Caire, G.; Molisch, A.F. Quality-Aware Streaming and Scheduling for Device-to-Device Video Delivery. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2319–2331. [[CrossRef](#)]
37. Figueroa, R.L.; Zeng-Treitler, Q.; Kandula, S.; Ngo, L.H. Predicting Sample Size Required for Classification Performance. *BMC Med. Inform. Decis. Mak.* **2012**, *12*, 1–10. [[CrossRef](#)]
38. Liu, G.Y.; Chang, T.Y.; Chiang, Y.C.; Lin, P.C.; Mar, J. Path Loss Measurements of Indoor LTE System for the Internet of Things. *Appl. Sci.* **2017**, *7*, 537. [[CrossRef](#)]



Reproduced with permission of copyright owner. Further reproduction
prohibited without permission.