Adaptive client training scale orchestration for federated learning

Younghwan Jeong [1], Taewon Song [2] and Taeyoon Kim [3,*]
[1,3] Department of computer engineering, Dankook University, Yongin-si, Gyeonggi-do, 16890, Korea
E-mail:{cjstntjd123,2000kty}@dankook.ac.kr
[2] Department of Internet of Things, Soonchunhyang University, Asan 31538, Korea
E-mail:twsong@sch.ac.kr

***Abstract*—Federated learning (FL), in contrast to traditional centralized learning, has gained significant attention as it enables the training of high-performance neural networks by maintaining user data locally while exchanging model updates between the server and clients, i.e., end-to-end network exchange. However, in practical environment, when considering various anonymous participants' computational resources, only a minority of clients can comply with the constraints imposed during the server-side training process. To address this issue, we propose Adaptive Scaling-Federated Learning (AS-Fed) based on Deep Q-Network (DQN) to dynamically orchestrate client's local data, allowing the inclusion of a larger number of clients in the training procedure. Our experimental results demonstrate that the proposed AS-Fed approach outperforms the legacy scheme, achieving higher normalization performance during the training process.**

***Index Terms*—federated learning, adaptive scaling, normalization performance, Deep Q-network**

I. Introduction

THANKS to remarkable advancements in hardware and improvements in communication environments, in recent years, there has been a significant increase in the generation and storage of large-scale data from various edge devices such as smartphones, home appliances, and sensors. To effectively train high-performance neural network models that capture the unique characteristics of this massive device data, Federated Learning (FL) has been proposed [1], [2]. FL enables training models locally on heterogeneous devices by exchanging models between the server and clients, thus preventing the leakage of sensitive user data and achieving advantages such as reduced communication and storage costs.

Due to these benefits, there has been numerous research in recent years to expand FL into practical applications. McMahan et al. [3] proposed FedAVG, which improved the training efficiency of FL by performing multiple local updates on user devices. Li et al. [4] introduced FedProx, which applied a proximal term to aggregate partial tasks from diverse user devices. Chai et al. [5] mitigated training delays through the classification of user devices with similar responsiveness using Tier in FedAT. Xie et al. [6] proposed FedAsync to efficiently aggregate user devices in a non-IID environment. This approach normalizes staleness weights to control asynchronous noise, adapting parameters to achieve non-convex optimization. Furthermore, various research efforts have been conducted to overcome the limitations of heterogeneity, a major challenge in FL. [7]–[10]

However, despite these diverse efforts, existing approaches are fundamentally designed to exclude user devices that do not comply with the training constraints imposed by the server. This not only hinders the global model's normalization performance to train on a wider range of data but also results in wastage of user device resources.

To address these issues and maximize the participation of user devices in the training process, this paper proposes AS-Fed. AS-Fed analyzes the training constraints imposed by the server and the available resources of user devices using Deep Q-Network (DQN). In other words, the server coordinates whether or not user devices connected to the network participate in training, and when participating, the size of data to be trained. To evaluate the performance of AS-Fed, experiments are conducted on benchmark datasets under various training constraint environments. The results demonstrate that AS-Fed achieves higher normalization performance compared to other FL schemes aimed at improving training efficiency.

The remainder of this paper is as follows: Section 2 explains the system model of AS-Fed, Section 3 performs problem statement and implementation of AS-Fed, Section 4 presents the experimental results, and finally, Section 5 concludes the paper.

II. System model

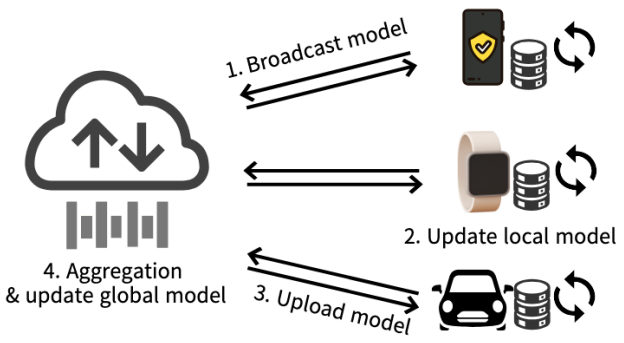

Fig. 1. Example of federated learning training sequence.

Figure 1 illustrates the training process of Federated Learning (FL). FL is divided into four sub-procedures, totally referred to as global iterations. Through the repetition of these

global iterations, the central server updates the global model. We assume an environment where diverse anonymous user devices (clients) are connected to the server. In the first procedure of the t-th global iteration, the central server randomly selects clients S_t to participate in training and broadcasts copies of the global model W_t to each client. In the second procedure, each participating client assigns the received copy of the global model as its local model. Then, each client performs local updates, following the training procedure defined by the central server, for a specified number of iterations using the local data they possess.

$$w_{t,k}^{i} := w_{t,k}^{i} - \eta \nabla F_i(w_{t,k}^{i}, D_i), k = 1, 2, \cdots, K \quad (1)$$

where, K is the number of local updates determined by the server, and $w_{t,k}^{i}$ is a i-th participating client's local model that performs the k-th local update in the t-th global iteration. η is the learning rate and D_i is the local data of i-th client.

In the third procedure, participating clients transmit their updated local models to the central server before a predetermined time limit expires. If a client fails to complete the second procedure within the time limit, it is automatically discarded and not aggregated. In the fourth procedure, the central server aggregates the updated models received from participating clients. Then, the global model is updated by performing a weighted sum proportional to the size of local data for each participating client as

$$W_{t+1} = \sum_{i=1}^{|S_t|} \frac{|D_i|}{D_t} W_t^i, \text{ where } \sum_{i=1}^{|S_t|} \frac{|D_i|}{D_t} = 1, \quad (2)$$

and where W_t^i is the updated local model of the i-th participating client. S_t is the set of participating clients in the t-th global iteration, and D_t is the sum of local data of all participating clients. Through the repetition of these procedures, the global model gradually converges to an optimal point.

However, in the FL training process where anonymous clients participate, only a minority of clients satisfy the two requirements imposed by the server: (1) performing a fixed number of local updates and (2) uploading the updated model to the central server within a specified time limit. This is because each participating client has different computational resources, communication conditions, and local data sizes. The existence of such heterogeneous clients can reduce the overall data size that the server can train on and can be a cause of performance degradation in the global model. Therefore, it is necessary to find an moderate approach that considers the diverse resource environments of heterogeneous clients while satisfying the constraints of the central server.

III. Adaptive Scale-Federated Learning (AS-Fed)

In this section, we propose AS-Fed (Adaptive Scale-Federated Learning) to maximize the participation of training participants. First, we present 1) the problem statement of AS-Fed and 2) describe its implementation.

A. Problem statement of AS-Fed

To enable heterogeneous training participants with limited resources on their respective devices to perform model training effectively, it is necessary to adaptively adjust the size of the training data used. Therefore, AS-Fed employs adaptive data scaling based on Deep Q-Network (DQN) to allow diverse participants to participate in the training procedure, despite the training constraints imposed by the server.

There are three main factors that influence the training time of multiple clients participating in the global model training of AS-Fed: 1) Training data size, 2) Computation time, and 3) Transmission time.

1) Training data size: Within each global iteration, the clients c_i belonging to the client set S_t selected by the server at the beginning of the t-th global iteration participate in the training using their respective local data. During the limited training time for each participant, the size of the data they will train on can be determined as follows:

$$D_{c_i} = K \times |d_{c_i}|, c_i \in S_t \quad (3)$$

where, K is the total number of training epochs limited by the server, and $|d_{c_i}|$ is the size of local data held by participating client c_i.

2) Computation time: Participating clients $c_i \in S_t$ participate in training with different computing resources. Accordingly, F_{c_i} is defined as the CPU frequency of the participating client. If the client consumes f_{c_i} to process one sample local data, the client c_i computation time $T_{c_i}^{Com}$ is expressed as:

$$T_{c_i}^{Com} = \frac{D_{c_i} \times f_{c_i}}{F_{c_i}} \quad (4)$$

3) Transmission time: The server-side training deadline includes the time for participating client $c_i \in S_t$ to perform local updates and then upload the updated model W_i^t to the server. Therefore, if the uplink frequency of the participating client is A_{c_i}, the transmission time $T_{c_i}^{Trans}$ of the client c_i is expressed as follow

$$T_{c_i}^{Trans} = \frac{|W_i^t|}{A_{c_i}} \quad (5)$$

where, $|W_i^t|$ is the parameter size of the local model updated by client c_i.

The total training time $T_{c_i}^{Train}$ taken by the participating client c_i for local update based on the above three factors is expressed as

$$T_{c_i}^{Train} = T_{c_i}^{Com} + T_{c_i}^{Trans} \leq T^{deadline} \quad (6)$$

When assuming that the availability of resources remains constant for participating clients throughout the training process, the factor that has the greatest impact on the overall training time $T_{c_i}^{Train}$ is the size of the training data $|d_{c_i}|$. Therefore, if the server can pre-determine the training data size of participating clients, even in a fixed training constraint environment, more clients can complete the training procedure while meeting the deadlines. In AS-Fed, the server also distributes a copy of the global model W_t along with the

scale coefficients determined via the DQN model. This allows clients to adjust the size of the local data used for training in heterogeneous resource environments.

B. Implementation of AS-Fed

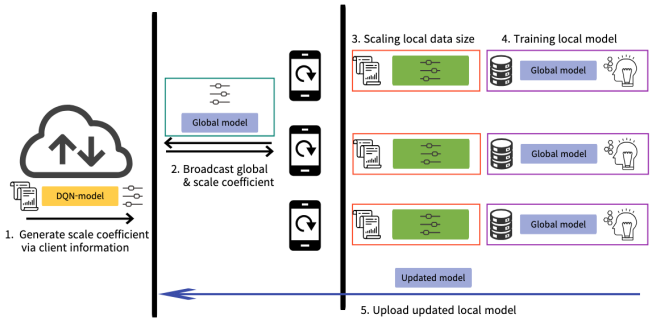

Fig. 2. Illustration of AS-Fed architecture.

Figure 2 illustrates the architecture of AS-Fed. In the conventional FL training procedure, N participating clients were randomly selected. However, in AS-Fed, the server prioritizes the selection of N clients from the network-connected clients using the DQN model. Simultaneously, the DQN model outputs scaling coefficients within the [0,1] range. Subsequently, the server sends these scaling coefficients along with a copy of the global model to the selected N clients. The clients utilize the received scaling coefficients to perform the training procedure using locally scaled data that matches the distribution of their current local data.

To train agent via DQN model, the server conducts offline training of the agent. The states, actions, and reward functions for training the agent to obtain the optimal scaling coefficient are defined as follows

States : The agent observes the information regarding the available resources of the current connected user device and generates the state s_{c_i} based on it. The available resource information of the participating client c_i is defined as $s_{c_i} =< |d_{c_i}|, F_{c_i}, A_{c_i}, \mu_{c_i}[t] >$. where, $\mu_{c_i}[t]$ is the aging term, which means the number of global iterations in which the client c_i is not selected by the server, and is expressed as

$$\mu_i[t+1] = (\mu_i[t]+1) \times (1 - i_{i,t}), i_{i,t} \in \{0,1\} \quad (7)$$

where, $i_{i,t}$ is an indicator variable indicating whether or not the server has selected.

Actions : To scale the size of the local data for the participating clients during training, scaling coefficients in the range of $a_{c_i} = [0,1]$ are defined. To prevent DQN model saturation caused by an excessive action space, the scaling coefficients are divided into increments of 0.1, ranging from 0 to 1.

Reward functions : The reward function is designed to optimize the selection of participating clients. The server-side DQN agent should coordinate as many participating clients as possible to participate in the training process, while each client trains on a larger number of local data. The reward function for this is expressed as

$$R_{c_i} = \begin{cases} min(\alpha, \frac{\mu_{c_i}}{\mu^{AVG}} \times \frac{T^{deadline} - T^{Trans}_{c_i}}{T^{Train}_{c_i} - T^{deadline}_{c_i}}), & \frac{T^{com}_{c_i}}{T^{deadline} - T^{Trans}_{c_i}} \leq 1, \\ -1, & \frac{T^{com}_{c_i}}{T^{deadline} - T^{Trans}_{c_i}} > 1, \\ 1, & a_{c_i} = 0. \end{cases} \quad (8)$$

where α is the maximum reward limit to avoid diverging the DQN model. μ^{AVG} is the average of μ_{c_i} of all clients connected to the network. Meanwhile, through a_{c_i} of the DQN model, the server can decide whether or not the client participates in training. If a_{c_i} is 0, the client is excluded from training. Accordingly, the DQN model performs an optimal choice between client exclusion and participation by some scaled local data. This allows the server to approximate $\hat{Q} : S \times A \rightarrow \mathbb{R}$ the parametric function to non-linear state-action pairs. Accordingly, the DQN training at each time step t is formulated as follow

$$L(\theta^{train}_t) = \sum_{<s,a,r,s'> \in D_b} (y^{DQN}_t - \hat{Q}(s,a;\theta^{train}_t))^2 \quad (9)$$

where y^{DQN}_t is expressed as

$$y^{DQN}_t = r_{t+1} + \gamma \max_{a' \in A} \hat{Q}(s', a'; \theta^{target}_t) \quad (10)$$

where θ^{train}_t is a set of weights that are updated at time step, and θ^{target}_t is a delayed weight that is copied at regular intervals to stabilize the training. Through this, the server can perform progressively optimized participating client selection.

IV. Performance Evaluation

A. Simulation setup

To evaluate the performance of AS-Fed, we assume the following experimental setup. The benchmark dataset used in the experiments is CIFAR-10, consisting of 10 classes with 5,000 training data and 1,000 test data per class. There are a total of 1,000 clients connected to the network. Each client possesses 200 local data, and the local data for each client follows an independent and identically distributed (IID) distribution with 20 uniformly sized samples per class. The sample data processing speed $\frac{F_{c_i}}{f_{c_i}}$, based on the CPU frequency of each client, takes random values between 1 and 30. The parameter $T^{Trans}_{c_i}$ for participating client c_i is randomly chosen between 1 and 10. In each global iteration, 20 clients are selected to participate in the training procedure. Additionally, each client performs $K = 5$ local updates in one global iteration. The training deadline T is set as the average of the mean training time for all clients, normalized to 1.

B. Impact of AS-Fed

In this section, we evaluate the training performance of AS-Fed based on the training deadline. Figure 3a presents a performance comparison when the deadline T is set to 1. AS-Fed demonstrates higher accuracy and faster convergence

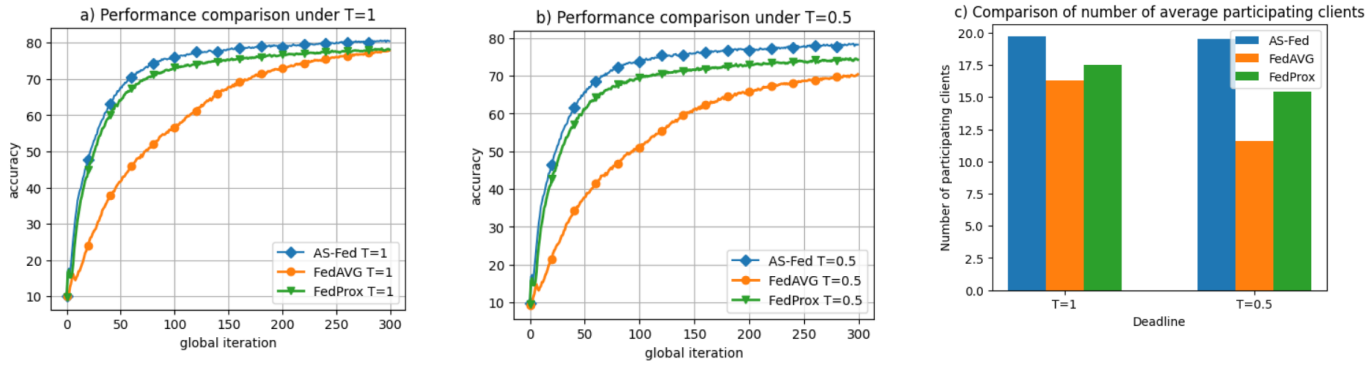

Fig. 3. Test set accuracy vs. global iteration for Cifar-10 under $T = [1, 0.5]$ setting

compared to other existing FL schemes such as FedAVG and FedProx. Specifically, AS-Fed achieves accuracy improvements of [2.47%, 2.33%] over FedAVG and FedProx, respectively. This can be attributed to AS-Fed's aggregation of more clients in the training procedure through training data scaling in each global iteration and the utilization of fresher training data through the aging term $\mu_{c_i}[t]$.

Figure 3b shows the performance comparison when the deadline T is reduced to 0.5. As observed from the figure, both FedAVG and FedProx experience accuracy degradation compared to the $T = 1$ scenario. Specifically, FedAVG and FedProx suffer accuracy reductions of [7.68%, 3.78%], respectively. This can be attributed to a larger number of clients failing to process the server's training constraint and being dropped. Consequently, the global model is trained with fewer data, leading to a decrease in normalization performance. On the other hand, as depicted in Figure 3b, 3c, AS-Fed adapts its local data size through training data scaling, resulting in minimal changes in the number of dropped clients even with the reduction in T. Therefore, AS-Fed effectively trains with a larger number of unique local data, leading to a smaller accuracy drop of 1.79% compared to other FL schemes. These experimental results demonstrate that AS-Fed successfully performs training in the heterogeneous resource environment of FL through adaptive training data scaling.

V. Conclusions and Future Works

In this paper, we propose AS-Fed, a method for adaptively scaling training data in the heterogeneous resource environment of FL. AS-Fed leverages a server-side DQN model to select participating clients and adjust the training data size using scaling coefficients. This enables a diverse set of clients to overcome server-side training constraints and participate in the training procedure, providing the global model with opportunities to train on more unique local data. Experimental results demonstrate that AS-Fed achieves higher test accuracy compared to legacy FL schemes, showcasing its effectiveness in maintaining normalization performance. For future work, we will explore client-driven resource allocation in dynamically changing resource environments of FL, aiming to effectively train models even in scenarios where user devices exhibit dynamic resource availability.

Acknowledgment

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2022R1F1A1076069).

References

[1] K. Jakub, B. McMahan, R. Daniel, "Federated optimization: Distributed machine learning for on-device intelligence". *arXiv 2016*, arXiv:1610.02527.

[2] K. Peter, B. McMahan, A. Brendan, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.* vol. 14, pp. 1-210, 2021

[3] B. McMahan, M. Eider, R. Daniel, H. Seth, A.A. Blaise, "Communication-efficient learning of deep networks from decentralized data," In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)* Fort Lauderdale, FL, USA, 9–11 May 2017.

[4] T. Li, A.K. Sahu, M. Sanjabi, M. Zaheer, A. Talwalker, V. Smith, "Federated optimization in heterogeneous networks," *Proc. Mach. Learn. Syst.,* vol. 2, pp. 429-450, 2020

[5] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, H. Rangwala, "Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data." *arXiv 2020*, arXiv:2010.05958.

[6] C. Xie, S. Koyejo, I. Gupta, "Asynchronous federated optimization." *arXiv 2019*, arXiv:1903.03934.

[7] M. Mohri, G. Sivek, A.T. Suresh, "Agnostic Federated Learning," In *Proceedings of the 36th International Conference on Machine Learning* Long Beach, CA, USA, 9–15 June 2019.

[8] F. Zhou, G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI* ,Sweden, 13–19 July 2018.

[9] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, Y. Cheng, "TiFL: A tier-based federated learning system," In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing,*Stockholm, Sweden, 23–26 June 2020, pp. 125-136

[10] F. Zhou, G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization." In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence,* Stockholm, Sweden, 13–19 July 2018.