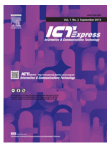
Available online at www.sciencedirect.com

ScienceDirect

ICT Express 10 (2024) 863–870



www.elsevier.com/locate/icte

DRL-empowered joint batch size and weighted aggregation adjustment mechanism for federated learning on non-IID data

Juneseok Bang[a], Sungpil Woo[b], Joohyung Lee[a,*]

[a] *The Department of Computing, Gachon University, Seongnam, Republic of Korea*
[b] *Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea*

Received 4 January 2024; received in revised form 15 March 2024; accepted 28 April 2024
Available online 30 April 2024

Abstract

To address the accuracy degradation as well as prolonged convergence time due to the inherent data heterogeneity among end-devices in federated learning (FL), we introduce the joint batch size and weighted aggregation adjustment problem, which is non-convex problem. To adjust optimal hyperparameters, we develop deep reinforcement learning (DRL) to empower a mechanism known as Batch size and Weighted aggregation Adjustment (BWA). Experimental evaluation demonstrates that BWA not only outperforms methods optimized solely from either a local training or server perspective but also achieves higher accuracy, with an increase of up to 5.53% compared to FedAvg, and additionally accelerates convergence speeds.

Keywords: Data heterogeneity; Deep reinforcement learning; Federated learning

1. Introduction

In the rapidly evolving world of various end-devices such as mobile phones, wearables, and autonomous vehicles, which produce large amounts of data, the combination with advancing artificial intelligence (AI) technologies holds great promise for data analysis. Given the distributed nature of these devices and the increasing emphasis on data privacy, an ideal scenario emerges for adopting federated learning (FL). FL allows individual end-devices to collectively build a global model by exchanging locally trained machine learning (ML) model parameters, rather than actual local data, ensuring privacy and efficiency [1,2]. However, achieving comparable training efficiency (e.g., convergence and accuracy) in FL compared to traditional centralized learning is more challenging due to the data heterogeneity of distributed end-devices. For example, when these devices have non-identically and independently distributed (non-IID) data, the training efficiency of typical stochastic gradient descent (SGD) based algorithms tends to decrease.

Numerous studies have attempted to enhance FL performance in non-IID environments by focusing on (i) local model updates and (ii) aggregation methods. Regarding local model updates, various strategies have been developed to address ‘client-drift’ due to data heterogeneity during these updates. These include aligning the local model more closely with the latest global model, introducing regularization to reduce directional inconsistencies, and adjusting the batch size across clients etc., [3–5]. From the aggregation perspective, different weighted aggregation techniques, which vary the weights assigned to contributions from end devices (e.g., FedAvg, FedCLS) [1,6], and optimizing client selection [7,8] have been explored.

Recently, due to unawareness of the explicit data distribution in each end-device, the application of deep reinforcement learning (DRL) has been explored to enhance decision-making in the complex and dynamic FL environment. Zhang et al. [5] used DRL to adjust local device batch sizes, demonstrating its impact on global model performance. Pang et al. [9] utilized DRL for clustering IID devices in FL without accessing local data. However, to the best of our knowledge, the integration of DRL in both local model updates and aggregation for training efficiency is still in its infancy. These two areas have been investigated separately, often leading to suboptimal performance. According to our motivation experiments, we

* Corresponding author.
E-mail addresses: dipsy1234@gachon.ac.kr (J. Bang), woosungpil@etri.re.kr (S. Woo), j17.lee@gachon.ac.kr (J. Lee).

https://doi.org/10.1016/j.icte.2024.04.011

found that the global model accuracy is significantly affected by batch size and aggregation weights in various FL settings and dataset. More specifically, the setting of fixed parameters causes two following problems. First, an optimal batch size exists that varies depending on the dataset or different FL settings. However, the local training process in most of the existing works fails to consider this by using a fixed batch size. Second, from the server perspective, challenges arise when the aggregation weights are biased towards certain local models. This bias can reduce the overall generalization ability of the global model. Therefore, there is a significant opportunity to explore the joint optimization of these two aspects using DRL in the complex and dynamic FL process.

In this article, we design a novel DRL-empowered Batch size and Weighted aggregation Adjustment (BWA) mechanism for FL on non-IID data environments. Our approach, distinct from existing methods, leverages a DRL algorithm to simultaneously determine two key aspects: (i) the optimal batch size for local training, and (ii) the aggregation weights for each device to expedite the update of the aggregated model. Additionally, the BWA mechanism is designed to automatically adjust to various FL settings, dynamically tuning hyperparameters in response to the level of data heterogeneity. This removes the need for meticulous hyperparameter selection for different scenarios. Our contributions can be summarized as follows.

- In our preliminary experiment, we investigate the impact of hyperparameters including batch size and weighted aggregation on the accuracy performance of the global model. Our findings demonstrate that optimal hyperparameters depend upon the data distribution of end-devices.
- We formulate the joint batch size and weighted aggregation adjustment problem to maximize convergence accuracy by considering both the server-side and local training-side aspects. To solve the non-convex problem, we design Markov decision process (MDP) using the score (e.g., model accuracy, loss, and weights divergence) of end-devices collected by local models and solve it by a DRL-based adjustment mechanism.
- We design DRL-empowered Batch size and Weighted aggregation Adjustment (BWA) mechanism, which is based on the Proximal Policy Optimization (PPO) DRL method, to select suitable batch size and allocate aggregation weights. Specifically, this DRL method is designed to enhance convergence accuracy while also accelerating convergence speeds. To achieve this, the improvement in accuracy per round is utilized as the reward function in the DRL method.
- The performance of BWA results on various datasets under different heterogeneity settings. The experiment demonstrates that BWA not only outperforms methods optimized solely from either a local training or server perspective but also achieves higher accuracy, with an increase of up to 5.53% compared to FedAvg, and additionally accelerates convergence speeds by up to 57.78% compared to FedAvg.

2. Related work and motivating example

2.1. Related work

Expanding on distributed training, with the consideration of privacy protection, McMahan et al. [1] introduced the concept of FL. It has attracted interest as a privacy-preserving, decentralized collaborative ML methodology. However, it has been demonstrated that the data heterogeneity among edge devices leads to a decline in performance [10,11]. To address this issue, existing methods primarily approach two mainstreams: optimizing local training on each devices and the aggregation phase on the server. With a method that considers the training direction of local models, in FedProx [3], a regularization is introduced to the local loss function, allowing local parameters to remain close to the global parameters. FedCos [4] mitigates directional inconsistencies among local models by adding a cosine normalization term. Zhang et al. [5] improve the performance of local training by adjusting the batch size of local devices, and verify that the batch size of local devices affects the performance of the global model. However, from a server-side perspective, given that the aggregation weight is biased, the global model may suffer from slow convergence or poor inference performance.

Another way to tackle the non-IID challenge is to adjust global model during the process of aggregation. FedAvg [1], a widely adopted aggregation algorithm, updates the global model by allocating large weights to devices with larger dataset. FedCLS [6] calculates a device's aggregation weight by considering both the number of classes in its labels and the quantity of data samples. Automated tuning of hyperparameters is feasible with grid and random search methods [12], yet they often consume considerable time by frequently exploring ineffective regions within the search space. For advanced random search [13] and Bayesian optimization [14], assessing the suitability of hyperparameter settings still requires multiple training runs. In FL environments, especially with complex training models (e.g., deep learning models) and when client data information is inaccessible, conducting numerous training iterations without access to clients' information is not feasible due to the limited computational resources and lack of information in real-world FL scenarios. Auto-FedAvg [15] and FedLAW [16] try to learn the aggregation weights on given dataset by gradient descent. Several approaches apply DRL to adjust aggregation weights [9,17]. However, as the aforementioned solutions not enhancing client-side training efficiency but focus on optimizing server-side aggregation, they typically achieve sub-optimal performance.

2.2. Motivating example

Although there are results showing that the performance (e.g., accuracy and convergence) of deep neural networks in the centralized ML varies depending on the batch size, as in [18,19], limited empirical studies have been conducted in FL, which is a different training paradigm from centralized ML. Furthermore, as we discussed, adjusting the aggregation

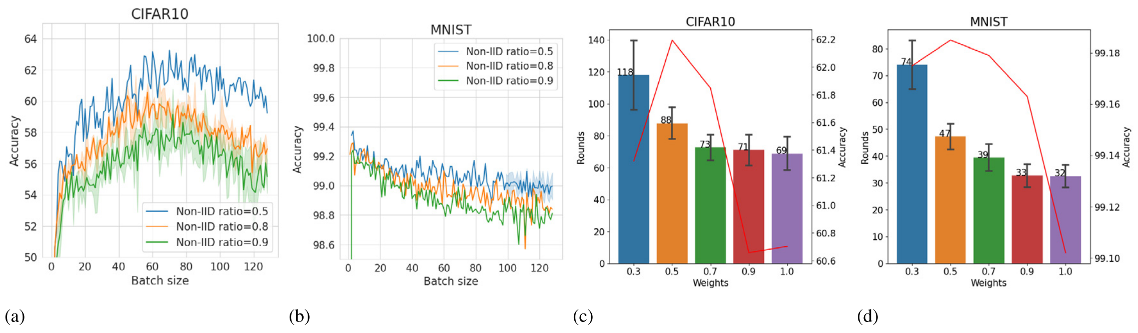


Fig. 1. Test accuracy versus communication rounds on CIFAR10 and MNIST with varying batch sizes (a), (b) and aggregation weights (c), (d). In (c) and (d), where the ratio of IID to Non-IID devices is equal, the target accuracy for CIFAR10 and MNIST is set at 60% and 99%, respectively.

weights for the global model update has a significant impact on accuracy and convergence speed.

Hence, to emphasize the significance of our proposed work, based on the preliminary simulation result as depicted in Fig. 1, we will highlight the critical role of adjusting both batch size and aggregation weights in enhancing the efficiency of FL. This example utilizes the MNIST and CIFAR10 datasets, examining both IID and non-IID scenarios. In Fig. 1(a) and (b), we observe a significant impact of batch size on overall performance, varying with the ratio of non-IID devices to the total number of devices. It also demonstrates the existence of an optimal batch size, with distinct trend shapes in different datasets. These findings emphasize the complexity of identifying the optimal batch size in varied and dynamic FL environments, such as those involving different types of datasets and a proportion of non-IID devices. The Fig. 1(c) and (d) illustrate the rounds needed to achieve the target accuracy and the maximum accuracy under varying aggregation weights for IID devices. It indicates that allocating biased weights to IID devices results in a failure to attain high maximum accuracy, despite achieving the target accuracy rapidly. This is because if an IID device is assigned biased weights, the information from local models of non-IID devices may not effectively contribute to the global model, leading to a hindrance in generalization ability and an overall degradation in performance. In the following section, we investigate joint batch size and aggregation weights adjustment for FL without accessing local datasets by employing DRL.

3. System model and problem formulation

3.1. Deep reinforcement learning

The fundamental principle of reinforcement learning entails the agent's exploration of optimal actions to attain a long-term optimum through iterative interactions with the environment [20]. The process of interaction can be represented using a Markov Decision Process (MDP). The MDP is defined by the tuple {S, A, P, R}. In this representation, S denotes a set of states s, and A denotes a set of actions a. The state transition function, denoted as $P : S \times A \times S \rightarrow [0, 1]$, calculates the probability of transitioning from the current state s_t to the next state s_{t+1} given the current action a_t. The reward at global round t, denoted as r_t, is determined by the reward

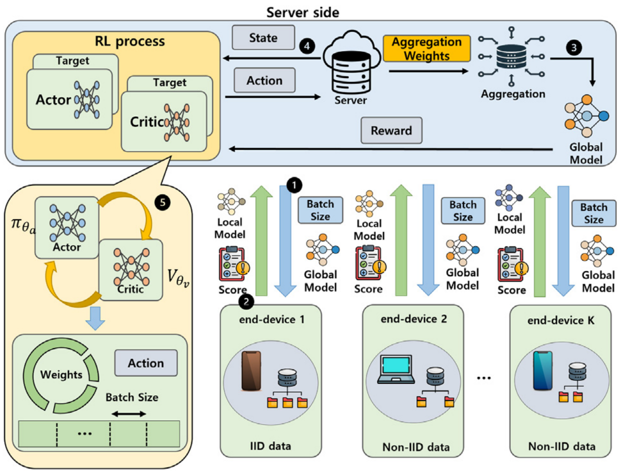


Fig. 2. The structure of BWA mechanism in FL. In the figure, each number corresponds to a specific step: 1. distributing actions and the global model, 2. local training by batch size, 3. updating global model by aggregation weights, 4. collecting state, and 5. DRL-based choosing actions, respectively.

function $R : S \times A \rightarrow \mathbb{R}$. Additionally, future rewards are discounted by a factor γ in the range $[0, 1]$. The expected discounted cumulative reward is commonly formalized as the value function through the Bellman equation: $V^{\pi}(s) = \mathbb{E}_p\left[r_1 + \gamma \sum_{s_\tau \in S} P(s_t|s_1, a_1)V^{\pi}(s_\tau)\right]$, where τ is a sequence of transition. In the MDP, the objective is to identify an optimal policy $\pi^*(a|s)$ that dictates the chosen action a given state s, aiming to maximize the expected cumulative reward for the agent. Thus, the optimal state-value function can be derived by

$$V^*(s) = \max_{\pi} V^{\pi}(s), \quad \forall s \in S \tag{1}$$

3.2. System model

Fig. 2 depicts the overall architecture and workflow of our proposed approach, which mainly consists of three parts: (1) local training part with batch size; (2) global model updating part with aggregation weights; and (3) reward-based policy update part. Our BWA method involves following five steps.

Step 1 (Model and Actions Dispatching): At the beginning of each training round, the FL server decides the training task, such as target application, and actions (i.e., batch size and aggregation weights). The FL server, then, dispatches global model and batch size to the devices.

Step 2 (Local Training by Batch Size): Consider the set $\mathbf{K} = \{1, 2, \ldots, K\}$ representing a group of K end-devices (called as clients). Each client k in the set $\mathbf{K}$ has a private dataset denoted as D_k with a size of $|D_k|$ samples, where each data sample i in D_k can be expressed as $\xi_i = \{\mathbf{x}_i, y_i\}$, with vector $\mathbf{x_i}$ representing the features of the ith data sample and y_i being its corresponding label. In each communication round t, the local model initializes w_k^t, which is set to the global model received. Client k samples a batch set B_k^t from its training data with a batch size b^t. The client then updates its own model using the mini-batch stochastic gradient descent (SGD) method, expressed as

$$w_k^{t,s+1} = w_k^{t,s} - \eta \nabla g_k^{t,s}, \text{ where } g_k^{t,s} = \frac{1}{b^t} \sum_{i \in B_k^{t,s}} \nabla l_k(w_k^{t,s}, \xi_i), \tag{2}$$

where s and η are the SGD step and the learning rate, respectively. The batch size is equally used across all clients. The loss function L_k^t of client k is expressed by the equation:

$$L_k^t(w_k^t, b^t) = \frac{1}{|D_k|} \sum_{B_k^t \subseteq D_k} l_k(w_k^t, b^t; B_k^t), \tag{3}$$

After training local model, each client k uploads local model w_k^{t+1} and local score, which is used to update policy (details provided in Step 4), to the FL server.

Step 3 (Aggregation Weights Averaging): The global model is updated based on the assigned aggregation weights for each round, as indicated below:

$$w_0^{t+1} = \sum_{k \in K} \lambda_k^t w_k^{t+1}, \tag{4}$$

where w_0 and λ_k are the global model and aggregation weights of each client, respectively.

Step 4 (Collecting Score): In each round, to evaluate the effectiveness of chosen batch size and aggregation weights, the server leverages three scores, which include weights divergence, loss, and accuracy, which are collected by local models. These are used for the operation of BWA mechanism as states and reward. We employ weights divergence as the magnitude of model updates.

$$\|d_j\| = \|w_j^{t+1} - w_0^t\|, \, j \in 0, 1, \ldots, K, \tag{5}$$

where j indicates the server (when $j = 0$) and clients (excluding $j = 0$). The weights divergence is calculated by Euclidean norm. Loss is computed in Step 2, and accuracy is formulated as

$$\phi^t = \begin{cases} \phi_0^t(w_0^t, \lambda_k^t), & \text{if global model} \\ \phi_k^t(w_k^t, b^t), & \text{otherwise} \end{cases}, \forall k \in \mathbf{K}, \tag{6}$$

where ϕ^t represents accuracy for global and clients' models.

Step 5 (DRL-based Choosing Actions): The FL server acts as an agent, which trains using DRL algorithm to find the optimal batch size and aggregation weights (details provided in Section 4).

BWA repeats the above five steps continuously until the convergence of global model.

3.3. Problem formulation

Based on the five steps defined in the proposed BWA, it finds suitable batch size and aggregation weights jointly to maximize the global model accuracy. Then, the optimization problem can be formulated as

$$\mathbf{P0} : \max_{b^t, \lambda_k^t} \sum_{t}^{T} \alpha^{\phi_0^t}, \tag{7}$$

$$\begin{aligned} \text{subject to } & \text{C1: } T \leq \Gamma, \\ & \text{C2: } k \in \mathbf{K}, \\ & \text{C3: } b^t \in [1, 64], \\ & \text{C4: } \lambda_k^t \in [0, 1], \forall k \in \mathbf{K}, \\ & \text{C5: } \sum_{k}^{K} \lambda_k^t = 1, \forall k \in \mathbf{K}, \end{aligned}$$

The objective is to maximize the global model accuracy, where α is a positive constant greater than 1. As the rounds progress, **P0** is expected to exponentially increase with the growing accuracy. (C1) represents the maximal rounds Γ. (C2) indicates that k belongs to the set of clients. (C3) defines the range of batch size. (C4) specifies that for each client k in the set K, the λ_k^t should be within the range [0, 1], and (C5) states that the sum of λ_k^t across all clients in the set K must equal 1. **P0** is non-convex because accurately modeling the global model's accuracy as a convex function with respect to batch size and aggregation weights presents significant challenges. Therefore, in order to address the non-convex problem **P0** and leverage the self-learning capability of reinforcement learning, we employ DRL to adaptively adjust both the batch size and aggregation weights. Specifically, DRL has great potential for solving problems where the optimal solution is unknown or very complex, especially within FL faced with dynamic and diverse environments [5]. This approach overcomes the inefficiencies and limitations associated with traditional HPO methods by leveraging the learning capabilities of DRL to adapt to changes in the environment or data distribution effectively.

4. DRL-based joint batch size and aggregation weights selection

We apply the MDP to solve **P0**. In the BWA mechanism, the FL server, acting as the agent, utilizes the Proximal Policy Optimization (PPO) approach [21] within DRL, which is one of the policy optimization algorithms that use the actor–critic method. PPO's implementation of the actor–critic method allows for the simultaneous updates of the policy (actor) based on the rewards received, and the value (critic), assessing the chosen actions' effectiveness. PPO aims to strike an equilibrium among implementation simplicity, sampling efficiency, and tuning convenience. Its objective is to compute updates that maximize the expected returns (cumulative rewards) while ensuring a subtle deviation from the previous policy. To prevent the policy from updating too drastically, [21] introduced clipped term. This innovative approach of using a clipped term is what distinguishes PPO from other policy optimization algorithms. By utilizing PPO, BWA addresses the adaptive control problem of batch size and aggregation weights in FL, taking into account various FL settings.

4.1. MDP design

- State S: At tth global round, the state consists of: loss $L(w)$, accuracy ϕ^t, and the weights divergence $\|d\|$. Examining the accuracy and loss values of each model allows agent to gain insights into the distribution of data across clients. Additionally, the distance from the global model in the previous round provides information on how much the models deviate from it [22]. This comprehensive information takes into account both local and global models. Therefore, the state observed by the agent at global round t is represented by a $s^t = \{\{L(w_j)\}, \{\phi_j\}, \|d_j\|\}$, where $j \in 0, 1, \dots, K$.
- Action A: The action of the agent is determined batch size b^t, which is employed uniformly for training local models, and the aggregation weights of each client λ_k^t for updating the global model. Thus, the action space can be defined as $a^t = \{b^t, \{\lambda_k^t\}_{k \in K}\}$
- Reward R: Considering the goal of maximizing accuracy, we define the agent's reward function based on the relative increase in accuracy as follows.

$$r^t = \frac{\phi_0^{t+1} - \phi_0^t}{\phi_0^t}. \tag{8}$$

When the accuracy in the $(t+1)$th round increases compared to that of the tth round, the reward is positive; otherwise, the reward is negative. Utilizing the relative accuracy increase compared to the previous round ensures stability, as accuracy may vary significantly depending on the initial performance of different models. If ϕ_0^t achieves maximum accuracy during training, multiply r^t by the current round t to expedite the achievement of maximum accuracy.

4.2. BWA algorithm details

Algorithm 1 summarizes the BWA algorithm, which is the process of training the PPO networks for batch size and aggregation weights adjustment. We first initialize θ_a and θ_v are parameters of the actor network $\pi_{\theta_a}(a^t|s^t)$ and critic network $V_{\theta_v}(s^t)$, respectively. $V_{\theta_v}(s^t)$ expects the gain of model accuracy in s^t. In steps 4–5, in each global iteration, we initialize the global model and the state. Step 7 indicates that, in each round t, the PPO network generates the action based on the current state s^t using the current policy. The action implies batch size b^t and the aggregation weights assigned to each client λ_k. In steps 8–15, each client trains local model with batch size based on (2) and send the updated local models and their scores to obtain next state s^{t+1}. FL server aggregates the local models based on (4), then, it calculates the reward according to (8). The MDP pair $< s^t, a^t, s^{t+1}, r^t >$ is stored

Algorithm 1: DRL-Based BWA Algorithm.

```
1:  Initialize Actor network and Critic network with $\theta_a$ and $\theta_v$
2:  Initialize the experience buffer $\mathcal{D}$
3:  $\theta_a^{old} \leftarrow \theta_a$
4:  for each global iteration do
5:      Initialize the model weight $w_0^1$ and $s^1$
6:      for each round $t$ = 1,2,...,T do
7:          Input $s^t$ to the policy network $\pi_{\theta_a^{old}}(a^t|s^t)$ to derive
            the action $a^t$
8:          for $k$ = 1,2,..., K do
9:              Client $k$ download parameter $w_0^t$
10:             Clients train the local models with batch size $b^t$
                and calculate score
11:             FL server updates the global model with $\lambda_k$
12:             Obtain next state
                $s^{t+1} \leftarrow \{\{L^{t+1}(w)\}, \{\phi^{t+1}\}, ||d^{t+1}||\}$
13:             Calculate the reward according to Eq. (8).
14:             Store the experience $(s^t, a^t, s^{t+1}, r^t)$ in $\mathcal{D}$
15:         end for
16:         for $m$ = 1,2,...,$M$ do
17:             Update the actor network $\theta_a$ with the experience
                data using PPO
18:             Update the critic network $\theta_v$ with the experience
                data by minimizing the loss function:
19:             $F(\theta_v) = \sum_{i=1}^{B}(r_i + \gamma V_{\theta_v}(s_{i+1}) - V_{\theta_v}(s_i))^2$
20:         end for
21:     end for
22:     $\theta_a^{old} \leftarrow \theta_a$
23:     Clear the experience buffer $\mathcal{D}$
24: end for
```

in experience buffer $\mathcal{D}$. In steps 16–23, the DRL agent utilizes experiences from the current episode for training. Due to importance sampling techniques, these experiences are employed for training the DRL agent multiple times M. Initially, the actor network is updated using the PPO algorithm (line 17). Subsequently, the critic network is updated using SGD to minimize the loss function with experiences from $\mathcal{D}$ (lines 18–19). Following M iterations of learning from experiences in $\mathcal{D}$, the new parameters of the actor network θ_a are assigned to the policy $\pi_{\theta_a^{old}}$ for the subsequent sampling. Simultaneously, the experience buffer is cleared. Upon completion of the training procedure, the DRL agent converges, and the optimal policy π^* becomes deployable in a real FL system.

5. Evaluation

5.1. Experiment setup

The experiments are conducted on a PyTorch-based platform, where we initially train the DRL agent on diverse datasets to derive optimal policies. The agent decides action during each communication round until it reaches the maximum steps. Following this, DRL-based BWA is deployed on the FL server to adjust batch size and aggregation weights throughout the entire training process. Additionally, we propose BWA with Bayesian Optimization (BO) [14], denoted as BO-based BWA, where BO is one of the HPO algorithms, which iteratively learns multiple hyperparameters to find the optimal value of the objective function (7). We conducted 100 training iterations to adjust the batch size and aggregation weights.

Table 1
Test Accuracy (%) After Achieving Convergence.

	MNIST		FashionMNIST		CIFAR10	
Setting (σ)	0.5	1	0.5	1	0.5	1
FedAvg	99.09	98.56	86.09	76.81	66.28	60.49
Adaptive-B	99.14	98.57	86.51	77.35	66.91	60.96
Adaptive-AW	99.16	98.56	86.67	77.69	66.69	60.90
BO-Based BWA	99.19	98.63	87.02	77.88	66.72	62.51
DRL-Based BWA	**99.24**	**98.76**	**87.05**	**78.05**	**67.37**	**63.83**

Dataset and Heterogeneity Setting. To evaluate the performance of BWAs, we utilize three representative image classification datasets in both color and grayscale: (1) MNIST, a grayscale handwritten number dataset that consists of 60 K 28 $\times$ 28 training images across 10 classes; (2) FashionMNIST, a grayscale clothing dataset featuring a wide variety of patterns and shapes, containing 60 K 28 $\times$ 28 training images in 10 categories; and (3) CIFAR10, a colored dataset with multi-object scenes and various backgrounds, comprising 50 K 32 $\times$ 32 images across 10 classes. These datasets are chosen for their realistic scenarios and are commonly used in other FL studies to validate findings [5,22]. The distinct CNN models are employed to train these datasets according to the settings in [22]. Each dataset is divided among 10 clients, each belonging to distinct categories (e.g., IID data and non-IID data). Specifically, we use σ to denote how many proportions of clients belonging to non-IID data to compare the performance of BWA in various heterogeneous scenarios.

- **Totally Non-IID setting** ($\sigma = 1$) : All clients in the FL possess non-IID data. Regarding the degree of heterogeneity, each client's dataset comprises only 2 classes according to the settings in [5].
- **50% Non-IID setting** ($\sigma = 0.5$) : Within the FL clients, 50% have non-IID data, while the remainder have IID data consisting of 10 classes. This setting represents the scenario in the FL environment with the low degree of heterogeneity.

Benchmarks. To evaluate the performance of BWA, the following three benchmarks are compared.

- **FedAvg** [1]: FedAvg, the most basic FL framework algorithm, allocates aggregation weights based on the data size of each client, where $\lambda_k^t = (|D_k|/|D|)$. It uses a fixed batch size of 10.
- **Adaptive-B** [5]: Adaptive-B is an adaptive algorithm known for its ability to dynamically adjust the batch size in each round. It optimizes local training with a focus on the client-side and employs aggregation weights in the same way as FedAvg.
- **Adaptive-AW** [17]: Adaptive-AW is a specialized algorithm designed to adjust the aggregation weights of individual clients in each round, aiming to optimize global model updates. The batch size is fixed at 10.

The number of local epochs is set as 5 and learning rate is 0.01.

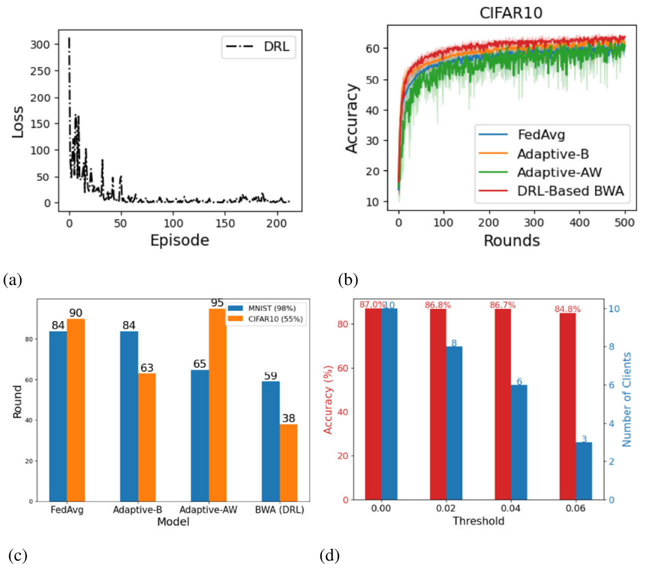


Fig. 3. Performance evaluations : (a) loss convergence of DRL agent; (b) test accuracy on CIFAR10; (c) required rounds for the given target accuracy with $\sigma = 1.0$; (d) client selection based on a specified threshold with $\sigma = 0.5$.

5.2. Experiment results

Training the DRL Agent. We first train the DRL agent, and test the performance of DRL training, including the training loss. The DRL agent training is conducted with 10 clients for 200 episodes. An episode starts with the initialization of the FL process and concludes once the FL completes the desired number of rounds. The PPO model in the DRL agent consists of 2 two-layer MLP networks, with 256 hidden states. Fig. 3(a) demonstrates the loss of the DRL agent on CIFAR10. The DRL agent achieved convergence by utilizing (8), which employs relative accuracy gains and ensures stability across various models.

Comparison of BWA with different methods. Fig. 3(b) compares the performance of the proposed method in CIFAR10 when σ is set to 1.0, which is the setting that represents the highest degree of heterogeneity and complexity images. BWA consistently achieves high accuracy from the initial rounds compared to other benchmarks. This indicates that additional rewards for attaining high accuracy are effective.

From Table 1, BWA achieves the highest accuracy across all σ settings. In CIFAR10, compared to FedAvg, as the σ value increases, the accuracy improvement with BWA is respectively 1.64%, and 5.53%. Moreover, as σ increases, a trend of decreasing accuracy is observed across all benchmark algorithms; however, it is notable that the performance degradation of BWA is relatively less pronounced. This reveals that BWA is robust under the high heterogeneity scenarios. Furthermore, as the FL environments become more heterogeneous, the two BWA methods achieve higher accuracy compared to approaches optimized solely from either a local training or server perspective. This suggests that jointly optimizing from both local training and server perspectives is more effective.

The accuracy of DRL-based BWA is slightly higher than that of BO-based BWA. This indicates that DRL-based approaches are more suitable in FL environments where information about the client's data distribution is inaccessible.

As the communication rounds accumulate, the communication overhead between the server and clients inevitably expands. Thus, it is crucial to achieve the target accuracy quickly. Fig. 3(c) shows the number of rounds needed to reach a target accuracy of 98% for MNIST and 55% for CIFAR10. DRL-based BWA reduced the number of rounds required to reach the target accuracy in CIFAR10 by approximately 57.78%, 39.68%, and 60.0% compared to FedAvg, Adaptive-B, and Adaptive-AW, respectively. DRL-based BWA provides efficiency and potential for practical application in FL scenarios.

Impact of Client Selection. Since the communication efficiency in FL is directly influenced by both the number of rounds and clients involved, we also conducted experiments on client selection as one of the important hyperparameters. Thus, we excluded clients whose aggregation weights fell below a certain threshold. As depicted in Fig. 3(d), the graph shows that at a threshold of 0, all 10 clients are selected, achieving an accuracy of 87.0%, while at a threshold of 0.06, the selected client count drops to 3 with an accuracy of 84.8%. Higher selected client counts lead to increased accuracy but also require more communication resources (e.g., radio bandwidth) for FL process. The experimental results suggest that with a threshold of 0.04, we can significantly increase the efficiency of communication while ensuring accuracy.

6. Discussion

Generalizability. Although training the DRL-based BWA algorithm is time-intensive, its compatibility with vanilla FL algorithms can be achieved with just a few simple modifications. This process involves merely transferring key metrics such as batch size, model accuracy, and loss as minor adjustments. Such streamlined modification underscores the ability of DRL-based BWA to be seamlessly integrated into existing FL frameworks, thereby enhancing their performance without the need for extensive modifications.

Considering Other Hyperparameters. In our study, we focused on the significant impact of batch size on local training and aggregation weights on global model updates. However, FL encompasses a wide range of hyperparameters, such as learning rate, local updates, and epochs, that also critically impact the FL process. Notably, the learning rate has a complex relationship with batch size [19], suggesting that a comprehensive examination of these hyperparameters together could further optimize FL strategies.

Lightweight Design and Cold start Problem. Training the DRL agent requires considerable time, which presents a significant challenge when used in real-time applications. The usability in real-time scenarios can be enhanced by making the DRL model more lightweight [23] or by innovatively combining it with other algorithms [24] to optimize the DRL model. Additionally, addressing the cold start problem that occurs during the initial stages of DRL model training [25] can also be considered a future research challenge.

7. Conclusion

This paper investigates the impact of hyperparameters, such as batch size and aggregation weights, on the accuracy of the global model. The findings highlight the importance of adjusting optimal hyperparameters based on the data distribution of end-devices. Our comprehensive experimental evaluation not only showcases BWA's enhanced robustness in non-IID environments but also its superiority in improving model accuracy and convergence rates. BWA outperformed methods that were optimized solely from either a local training or server perspective, achieving higher accuracy up to 5.53% compared to FedAvg. Furthermore, we have thoroughly discussed BWA's limitations and identified areas for future research.

CRediT authorship contribution statement

Juneseok Bang: Conceptualization, Data curation, Formal analysis, Investigation, Software, Writing – original draft. **Sungpil Woo:** Funding acquisition, Project administration, Validation. **Joohyung Lee:** Conceptualization, Supervision, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (2022-0-01032, Development of Collective Collaboration Intelligence Framework for Internet of Autonomous Things).

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: AISTATS, 2017, pp. 1273–1282.
[2] V. Gugueoth, S. Safavat, S. Shetty, Security of Internet of Things (IoT) using federated learning and deep learning-recent advancements, issues and prospects, ICT Express (2023).
[3] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proc. Mach. Learn. Syst. 2 (2020) 429–450.
[4] H. Zhang, T. Wu, S. Cheng, J. Liu, Fedcos: A scene-adaptive enhancement for federated learning, IEEE Internet Things J. 10 (5) (2022) 4545–4556.
[5] J. Zhang, S. Guo, Z. Qu, D. Zeng, Y. Zhan, Q. Liu, R. Akerkar, Adaptive federated learning on non-iid data with resource constraint, IEEE Trans. Comput. 71 (7) (2021) 1655–1667.
[6] D.M.S. Bhatti, H. Nam, FedCLS: Class-aware federated learning in a heterogeneous environment, IEEE Trans. Netw. Serv. Manag. (2023).
[7] W. Zhang, X. Wang, P. Zhou, W. Wu, X. Zhang, Client selection for federated learning with non-iid data in mobile edge computing, IEEE Access 9 (2021) 24462–24474.
[8] J. Zhao, X. Chang, Y. Feng, C.H. Liu, N. Liu, Participant selection for federated learning with heterogeneous data in intelligent transport system, IEEE Trans. Intell. Transp. Syst. 24 (1) (2022) 1106–1115.

[9] J. Pang, Y. Huang, Z. Xie, Q. Han, Z. Cai, Realizing the heterogeneity: A self-organized federated learning framework for IoT, IEEE Internet Things J. 8 (5) (2020) 3088–3098.
[10] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, 2018, arXiv preprint arXiv:1806.00582.
[11] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of FedAvg on non-iid data, 2019, arXiv preprint arXiv:1907.02189.
[12] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012).
[13] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, Adv. Neural Inf. Process. Syst. 24 (2011).
[14] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, S.-H. Deng, Hyperparameter optimization for machine learning models based on Bayesian optimization, J. Electron. Sci. Technol. 17 (1) (2019) 26–40.
[15] Y. Xia, et al., Auto-FedAvg: Learnable federated averaging for multi-institutional medical image segmentation, 2021, arXiv preprint arXiv:2104.10195.
[16] Z. Li, T. Lin, X. Shang, C. Wu, Revisiting weighted aggregation in federated learning with neural networks, 2023, arXiv preprint arXiv:2302.10911.
[17] P. Guo, et al., Auto-FedRL: Federated hyperparameter optimization for multi-institutional medical image segmentation, in: Eur. Conf. Comput. Vis., Springer, 2022, pp. 437–455.
[18] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Int. Conf. Mach. Learn., PMLR, 2013, pp. 1139–1147.
[19] S.L. Smith, P.-J. Kindermans, C. Ying, Q.V. Le, Don't decay the learning rate, increase the batch size, 2017, arXiv preprint arXiv:1711.00489.
[20] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[22] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: IEEE INFOCOM 2020-IEEE Conf. Comput. Commun., IEEE, 2020, pp. 1698–1707.
[23] S.S. Pandi, A. Senthilselvi, J. Gitanjali, K. ArivuSelvan, J. Gopal, J. Vellingiri, Rice plant disease classification using dilated convolutional neural network with global average pooling, Ecol. Model. 474 (2022) 110166.
[24] S.P. Sankareshwaran, G. Jayaraman, P. Muthukumar, A. Krishnan, Optimizing rice plant disease detection with crossover boosted artificial hummingbird algorithm based AX-RetinaNet, Environ. Monit. Assess. 195 (9) (2023) 1070.
[25] J. Lee, F. Solat, T.Y. Kim, H.V. Poor, Federated learning-empowered mobile network management for 5G and beyond networks: From access to core, IEEE Commun. Surv. Tutor. (2024).