JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 16, No. 1, March 2022, pp. 52-62

Stable Federated Learning with Dataset Condensation

Seong-Woong Kim and Dong-Wan Choi*
Department of Computer Science and Engineering, Inha University, Incheon, Korea
wauri6@gmail.com, dchoi@inha.ac.kr

Abstract
Federated learning (FL) is a new machine learning paradigm, where multiple clients learn their local models to collaboratively integrate into a single global model. Unlike centralized learning, the global model being integrated cannot be tested in FL as the server does not collect any data samples, further, the global model is often sent back and immediately applied to clients even at the middle of training such as *Gboard*. Therefore, if the performance of the global model is not stable, which is, unfortunately, the case in many FL scenarios with non-IID data, clients can be provided with an inaccurate model. This paper explores the main reason for this *training instability* of FL, that is, what we call *temporary imbalance* that happens across rounds, leading to loss of knowledge from previous rounds. To solve this problem, we propose a dataset condensation method to summarize the local data for each client without compromising on privacy. The condensed data are transmitted to the server with the local model and utilized by the server to ensure stable and consistent performance of the global model. Experimental results show that the global model not only achieves training stability but also exhibits a fast convergence speed.

Category: Smart and Intelligent Computing

Keywords: Deep learning; Federated learning; Dataset compression; Class imbalance

I. INTRODUCTION

Recently, deep learning has shown remarkable growth in many fields using large datasets, but training models on millions of data require enormous storage capacity and computational costs. In addition, existing centralized learning methods store and learn entire data in a data server, whereas in the real environment, data are generated or collected in several Internet of Things (IoT) devices, mobile devices, and autonomous vehicles, and as such exist in a distributed form. Therefore, collecting and storing distributed data in a server incurs significant collection costs and causes privacy problems. However, there is insufficient data on personal devices for application users to train their deep learning models. To solve this gap between the distributed data environment and centralized learning methods, "federated learning" [1-3], which uses distributed data to learn without transmitting clients' data, has emerged.

Federated learning is a machine learning paradigm that enables privacy protection by having clients train local models on their data, and a server to aggregate the trained local models. Thus, clients can have a model with good performance while protecting their personal information, and companies that operate parameter servers have the advantage of reducing data collection and computing costs required for model training. It can be claimed that federated learning is a win-win strategy that satisfies both service users and providers. However, unlike traditional centralized learning methods, which divide the entire data on the server into training, validation, and test datasets, the model is fully trained and evaluated (i.e., a pretrained model). In the federated learning scenario, data is unlikely to exist for validating the performance of

Open Access http://dx.doi.org/10.5626/JCSE.2022.16.1.52 http://jcse.kiise.org

Received 24 February 2022; **Accepted** 16 March 2022
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

the parameter server. Therefore, in federated learning, the performance of the model being trained cannot be estimated, making it difficult to optimize the hyperparameters. The unstable performance of a model can lead to various problems, such as deploying the erroneous model.

Existing federated learning methods place test datasets on the server and confirm model performance during the learning process. This is a non-realistic setting and is only possible in an experimental environment using public data. Alternatively, there is another way that prohibits particular clients from participating in the learning and makes them validation clients or generates validation data using some data from all clients [3]. However, this method is not practical if the number of participating clients is small or the amount of data is too less and cannot be divided for validation.

A sudden deterioration in model performance can cause serious problems when the model performance cannot be precisely evaluated. For example, in keyboard applications [4] such as next-word prediction or emoji prediction, poor performance only causes inconvenience to clients, but in medical fields such as brain tumor identification [5] or predicting the clinical outcome of patients with coronavirus disease 2019 (COVID-19) [6], rapid degradation of model performance can lead to more serious side effects. Therefore, the training stability of the federated learning model must be guaranteed.

The main problem with federated learning is data heterogeneity, where each client follows a different local data distribution (e.g., mobile users and hospitals). Due to data heterogeneity, the global model, which aggregates local models trained on heterogeneous distributions, suffers from poor performance and slow convergence [7]. In this paper, our focus is on the fact that a global model gets more unstable in the middle of training when data becomes more heterogeneous.

In Fig. 1, the parameter α controls the data distribution of the clients, and if α is small, the clients have a more biased data distribution. A detailed description of α is provided in Section V. The three graphs in Fig. 1 show that the higher the non-IID setting (i.e., the smaller α), the greater is the variation in model performance. Therefore, it is observed that non-IID not only impairs the overall performance or slows down the convergence rate of the model but also degrades the stability. As mentioned earlier, the instability problem is a serious disadvantage in federated learning, where the performance of the model cannot be validated, and the model needs to be deployed even in the middle of training. A more specific cause of the instability problem is that when collecting models from the server, only a fraction of the clients, participate in each round, and the distribution of heterogeneous data from participating clients causes class imbalance problems in that round. The analysis of the causes is discussed in more detail in Section III.

To address this challenge, previous works identified

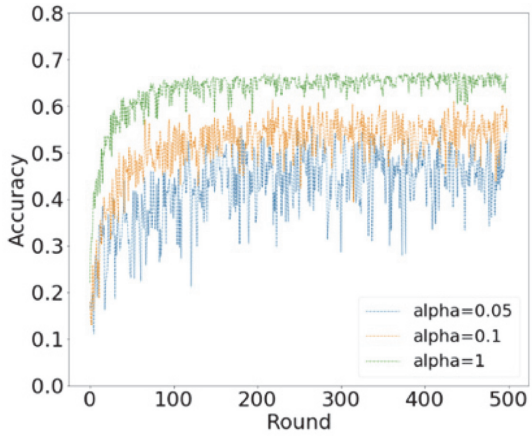

Fig. 1. Performance of federated learning depending on data heterogeneity on CIFAR-10.

the cause of performance decline as data heterogeneity in biased client model updates, which is referred to as "client drift." Research papers, attributing the problem to overall updates and out-of-line client updates, have proposed methods to limit the client updates [8-10]. However, because these methods aim to update the model of the previous round as much as possible, the model may continue to learn in the wrong direction if the learning efficiency is poor or if the update of the previous model is biased. Therefore, several studies have revealed that client update regularization theoretically alleviates heterogeneity problems, but does not work well in practical deep learning problems [11-13], especially when only some of the many clients participate in each round. We also confirm that these regularization-based methods do not work well through our experimental study. Another solution to data heterogeneity is to utilize a public dataset shared by all clients to prevent learning in extremely heterogeneous data situations; however, this is also an impractical method that does not fit the real world.

To match the imbalanced data distribution that varies each time, in this paper, the class data of each client are somehow undersampled and transmitted to the server. The server uses this class-balanced dataset to fine-tune the biased global model, thereby obtaining a more generalized model, ensuring training stability. However, privacy issues preclude personal data transfer to the server, and even if allowed, the randomness causes information loss during the selection of data samples. Therefore, we propose a method to minimize the loss of information and ensure privacy by compressing the original local data into an extremely smaller form by training to the point that we cannot recognize its original privacy information. Through experiments, it was demonstrated that the proposed method ensures faster convergence and more stable performance than other methods in various experimental settings. Furthermore, we also observe that the more heterogeneous the setting, the greater the gap is with the other algorithms.

II. RELATED WORK

A. Federated Learning

Federated learning was first proposed in [1] as a learning model for a distributed mobile device environment, and FedAvg was devised by applying the existing distributed stochastic gradient descent (SGD) algorithm. Research to improve the performance of federated learning algorithms can be divided into two categories: local regularization algorithms and alternative model aggregation algorithms. In local regularization, a loss term is added to limit the difference between local and global updates. FedProx [8] uses the proximal term, which is the difference between the parameters of the previous global model and the client model parameter, as a regularization term. SCAFFOLD [9] uses the difference between the entire local gradient direction (i.e., the global direction) and the local gradient direction as a regularization term. Moon [10] regularizes the model through contrastive learning by introducing the previous client model as a negative term and the global model as a positive term.

In the model aggregation improvement algorithm FedNova [11], it was determined that the general averaging method causes objective inconsistency, thus, a normalized averaging method was proposed. FedAdam [12] interprets the existing model aggregation algorithm as a pseudo gradient and applies the conventional adaptive optimization algorithm to server updates. In [13], an optimized aggregation method, FedMa, was proposed through neuron matching for each layer of the model, not just a weighted averaging of the model parameters.

B. Dataset Compression

Large datasets consisting of millions of samples are becoming the norm in deep learning models, but storing and using this scale of data for learning requires enormous costs and infrastructure. Accordingly, dataset compression has emerged, which ensures maximum performance of the learned model on the original dataset while reducing the size of the dataset. The first proposed data distillation (DD) method [14] learns the model using synthetic noise data and then optimizes synthetic data to minimize the learning loss of real data. Dataset condensation (DC) was proposed [15] as a training method to match the gradient when real and synthetic data are passed through the same model for the same problem. Differentiable Siamese augmentation (DSA) [16] further boosts the performance of DC by utilizing data augmentation techniques. In this work, DC [15] is exploited for federated learning to minimize the loss of information by compressing the original local data into synthetic data, which are not recognizable by humans, to ensure privacy protection.

C. Class Imbalance

In contrast with the experimental dataset, where the number of data samples in all classes is equal, real data distributions often have long-tails [17] with highly imbalanced volumes of class-wise data. This class imbalance problem causes the performance of the minority class to be extremely poor, compared to that of the majority class with a large number of data samples. Consequently, the overall performance is dominated by a particular class [18]. To solve the class imbalance problem, oversampling [19] and undersampling [20] methods train the model by rebalancing the data distribution. In addition, instead of the cross-entropy loss function, a class-balanced loss is used to assign different loss values to each class sample [21], and some approaches attempt to rebalance the distribution of the classifier [22].

In federated learning, there are some studies to address the overall class imbalance problem by considering the real heterogeneous data environment [23, 24]. However, in this paper, we solve the problem of class imbalance that more temporarily happens due to the randomly selected participating clients. The solution applies even when all the classes are evenly distributed globally.

III. ANALYSIS ON TRAINING INSTABILITY OF FEDERATED LEARNING

As mentioned in Section I, we first observe and analyze the fact that the performance of federated learning gets more unstable if the data becomes more heterogeneous. To this end, Fig. 2 presents two specific rounds (i.e., 120th and 121st) in our federated learning experiment of Fig. 1 to investigate why the performance gets most unstable when $\alpha = 0.05$ in Fig. 1. In particular, we focus on the 120th round (accuracy 43.9%) and 121st round (accuracy 21.3%), where the performance degradation of the FedAvg algorithm is noticeable.

First, we examine the round-wise class distribution. As the clients participating in each round are different, the round-wise class distribution can differ greatly, even between two consecutive rounds. This is a type of class imbalance problem that *temporarily* occurs when each client has a different distribution (i.e., non-IID), even if the classes are evenly distributed with respect to the entire dataset. The above *temporary imbalance* problem occurs as the number of data samples varies for each class and client. For example, in the 120th round, only three classes have more than 1,000 data samples, and the remaining seven classes have less than 400 data samples. In the 121st round, the number of data samples in the first class is over 4,000, but the other three classes have less than 50 samples. Since the performance of FedAvg is dominated by the majority class of each round, the overall performance becomes severely biased in the 121st

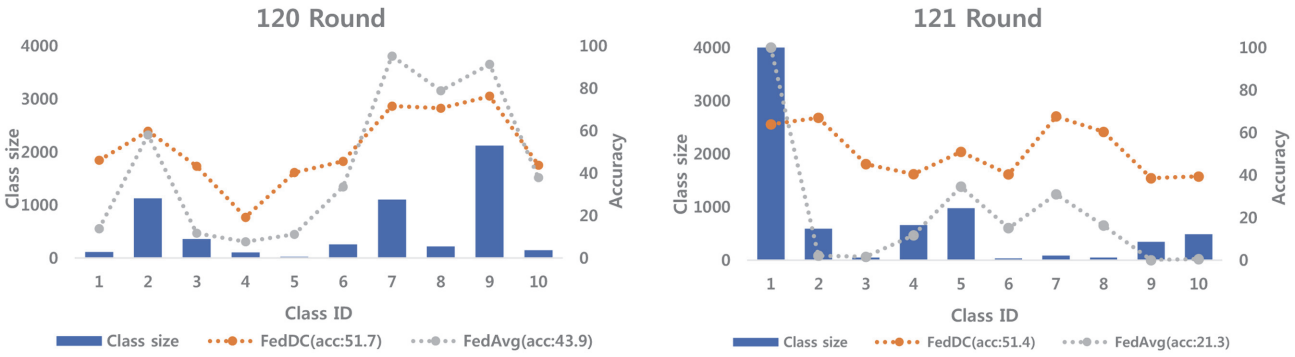

Fig. 2. Comparison of class-wise accuracy after learning (a) the 120th round and (b) 121st round, and the round-class distribution.

round than in the 120th round. This is because the first class in the 121st round, which has excessively more data than the other classes, leads to a global model biased to this majority class. Therefore, standard federated learning algorithms suffer from a catastrophic forgetting phenomenon of the knowledge learned in the previous round, which happens every round, resulting in training instability.

This paper proposes a solution to this temporary class imbalance problem in federated learning, called FedDC, which effectively alleviates such round-wise catastrophic forgetting. The basic idea of FedDC is that clients send a small amount of undersampled data for each class to the server along with the locally updated model. Thus, the server can receive data samples proportional to the number of classes from the clients, and therefore obtain a balanced dataset regardless of whether the client has imbalanced data. Subsequently, the generalization performance of the model can be restored by fine-tuning the global model with this balanced dataset. The challenge here is that we should not send raw data samples to the server due to privacy issues. Furthermore, even if we can, the performance of the global model can vary depending on which data samples are selected. Thus, the global model in this time can overfit the selected dataset, which may not represent the entire local data.

To achieve both stable performance and for privacy protection, FedDC compresses local data into a form of personal information not being revealed, and thereby the server receives the amount of compressed data proportional to the number of classes. With FedDC, a stable global model can be obtained by effectively rebalancing the round-wise class distribution with the compressed and balanced dataset in the server. As shown in Fig. 2, FedAvg and FedDC display similar performance in the 120th round, but FedDC improves the overall accuracy by fitting more minority classes well. In summary, unlike FedAVG, the accuracy of FedDC remains at 51.4% in the 121st round, that is approximately the same as the accuracy of 51.7% in the previous round.

IV. METHOD

In this section, we present FedDC, which is a balanced fine-tuning method that exploits a data compression scheme to solve round-wise class imbalance. General federated learning is described in Section IV-A, and the data compression techniques based on [15] are explained in Section IV-B. Based on these, the whole FedDC framework is finally proposed in Section IV-C.

A. The Framework of Federated Learning

In federated learning scenarios, the common objective is to minimize the following equation:

$$\min_{\theta} F(\theta) = \sum_{i=1}^{M} p_i (F_i(\theta) \coloneqq \mathbb{E}_{x\sim\mathcal{P}_i}[f_i(\theta; x)]) \quad (1)$$

$$\text{such that: } \sum_{i=1}^{M} p_i = 1 \text{ and } p_i \geq 0,$$

where $\theta \in R^d$ refers to the deep learning model, M is the number of clients, p_i is the weight assigned to the i^{th} client, F_i is the local objective function of the i^{th} client, and f_i is the loss function. $\mathcal{P}_i$ is the data distribution of the i^{th} client, which may be very diverse across clients.

Let $\mathcal{D}_i$ be the local dataset of the i^{th} client such that $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_M$ is the entire dataset from all clients. Then, the standard FedAvg algorithm uses the following equations to solve the optimization problem of Eq. (1).

$$F(\theta) = \sum_{i=1}^{M} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} F_i(\theta), \quad (2)$$

$$\text{where } F_i(\theta) = \frac{1}{|\mathcal{D}_i|} \sum_{x\in D_i} \ell_{ce}(\theta; x),$$

Thus, FedAvg assigns a weight for each client to be proportional to the amount of its local data. (i.e., $|\mathcal{D}_i|$). Then, the weighted average of the local parameters is computed, and finally the global parameters are obtained. $F_i(\theta)$ is the empirical risk minimization objective function; ℓ_{ce} is the cross-entropy loss commonly used in classification. Some variants of FedAVG add a regularization term to

ℓ_{ce} for the local model not to be far different from the global model.

The global model θ is trained for T rounds, and each round consists of the following three steps. First, the server broadcasts the global model to some randomly selected clients. Second, each selected client trains a local model with its own data, and then sends the trained model back to the server. Finally, the server updates the global model by aggregating the received local models. In the FedAvg framework, clients perform several mini-batch SGDs on their local model:

$$\theta_{i,k+1}^{t} = \theta_{i,k}^{t} - \eta \nabla \ell\left(\theta_{i,k}^{t}; x_{i,k}\right) \; for \; k \in [0, E-1], \quad (3)$$

where $x_{i,k}$ is the randomly selected data from $\mathcal{D}_i$, η is the learning rate, and $\theta_{i,k+1}^{i}$ is the local model updated by the i^{th} client after E iterations at the t^{th} round.

Even if the global model θ converges to the minima of (1), such a minima could be different from the counterpart obtained by a centralized model. This problem is mainly caused by data heterogeneity. In centralized training, the global model is trained by selecting (or generating) each batch under the IID assumption from the whole dataset $\mathcal{D} = \bigcup_{i=1}^{M} \mathcal{D}_i$. Thus, each mini-batch is drawn from the same distribution as the overall data distribution $\mathcal{P}$. Recently, deep learning models trained on these IID datasets (i.e., centralized training) perform even better than humans on the corresponding IID test data [25]. Contrary, in federated learning, it is assumed that the training dataset $\mathcal{D}_i$ of each client follows the client's own data distribution $\mathcal{P}_i$, which would not be the same as the overall data distribution $\mathcal{P}$. Therefore, each mini-batch of $\mathcal{D}_i$ is also assumed to be non-IID, meaning that its distribution does not follow $\mathcal{P}$. Consequently, the global model aggregated with local models $\sum_{i=1}^{M} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} F_i(\theta)$ does not have the same minima as that of the centralized model, which is trained under the IID assumption.

B. Dataset Condensation

To solve the problem mentioned in Sections III and IV-A, we propose a synthetic dataset learning method that summarizes and compresses the original dataset of clients.

Let $D_m = \{(x_n, y_n)\}|_{n=1}^{|D_m|}$ be the training dataset of the m^{th} client; the proposed dataset compression aims to obtain a synthetic dataset $S_m = \{(\tilde{x}_n, y_n)\}|_{n=1}^{|S_m|}$ smaller than the original dataset D_m. More formally, the goal of our dataset compression method can be represented as:

$$\mathbb{E}_{x \sim \mathcal{P}_m}[\ell(\theta^{D_m}; x)] = \mathbb{E}_{\tilde{x} \sim \mathcal{P}_m}[\ell(\theta^{S_m}; \tilde{x})] \quad (4)$$

$$\text{such that: } \theta^{D_m} = min_{\theta} \, \ell(\theta, x^*, y^*),$$

$$\theta^{S_m} = \min_{\theta} \ell(\theta, x^*, y^*), and \; |S_m| \ll |\mathcal{D}_m|,$$

where θ^{D_m} and θ^{S_m} denote the deep learning models trained on $\mathcal{D}_m$ and S_m, respectively. Our expectation here is that if the model θ^{S_m} is trained with much fewer data samples it can be generalized as θ^{D_m} in the real data distribution of $x^* \sim \mathcal{P}_m$. Note that the generalization performance is measured based on the hidden data x^*. Thus, to obtain a synthetic dataset satisfying Eq. (4), we minimize the gradient difference between the losses of synthetic and original data through the following training process [15].

$$min_{S_m} \, \mathbb{E}_{\theta \sim P_\theta}[D(\nabla_\theta F(\theta; S_m), \nabla_\theta F(\theta; \mathcal{D}_m))] \quad (5)$$

$$D\left(\nabla F_S(\theta), \nabla F_{\mathcal{D}}(\theta)\right) = \sum_{l=1}^{L} d\left(\nabla F_S\left(\theta^{(l)}\right), \nabla F_{\mathcal{D}}\left(\theta^{(l)}\right)\right) \quad (6)$$

$$d\left(\nabla F_S\left(\theta^{(l)}\right), \nabla F_{\mathcal{D}}\left(\theta^{(l)}\right)\right) = 1 - \frac{\nabla F_S(\theta^{(l)}) \cdot \nabla F_{\mathcal{D}}(\theta^{(l)})}{\|\nabla F_S(\theta^{(l)})\| \|\nabla F_{\mathcal{D}}(\theta^{(l)})\|},$$

where $\nabla_\theta F(\theta; S_m)$ and $\nabla_\theta F(\theta; \mathcal{D}_m)$ are the gradients of S_m and D_m, respectively, using a single model θ, which is randomly initialized to the distribution P_θ. We use (1-cosine similarity) for the matching loss function $D(\cdot,\cdot)$. The loss is calculated in a layer-wise manner, where $l \in [1, L]$ is the index of each layer, and $D(\nabla F_S, \nabla F_{\mathcal{D}})$ is a simple formulation of Eq. (5).

Algorithm 1. FedDC

Input: random initialize global model θ^0
for each round $t = 0,1, \ldots, T-1$ **do**
  Select a random subset $\mathbb{S}_t$ for the clients M
  Send the global model θ^t to $\mathbb{S}_t$
  for each client $i \in \mathbb{S}_t$ **in parallel do**
    $\theta_i^t \leftarrow$ **ClientUpdate**$(\theta^t, \mathcal{D}_i)$ using Eq. (3)
    $S_i \leftarrow$ **CondensedDataTraining**$(i, \mathcal{D}_i)$
    send the θ_i^t, S_i to server
  end for
  $\theta^t \leftarrow$ Aggregate the client models using Eq. (2)
  $\theta^t \leftarrow$ Finetune with S_i using Eq. (7)
end for
CondensedDataTraining$(i, \mathcal{D}_i)$
for $k = 0,1, \ldots, K-1$ **do**
  Initialize $\theta \sim P_\theta$
  for $c = 0,1, \ldots, C-1$ **do**
    Sample a minibatch $\mathcal{D}_i \sim B_{\mathcal{D}}^c$
    Compute $f_S(\theta) = \frac{1}{|S_i^c|} \sum_{x \in S_i^c} \ell_{ce}(\theta; x)$
    and $f_{\mathcal{D}}(\theta) = \frac{1}{|B_{\mathcal{D}}^c|} \sum_{x \in B_{\mathcal{D}}^c} \ell_{ce}(\theta; x)$
    Update $S_i^c = S_i^c - \eta \cdot \nabla D\left(\nabla F_s(\theta), \nabla F_{\mathcal{D}}(\theta)\right)$
  end for
end for
return condensed dataset S_i

C. FedDC: Stable Federated Learning

A detailed description of the proposed FedDC method is summarized in Algorithm 1. Basically, FedDC includes the following two additional processes to the three primitive steps of the standard federated learning method: (1) the data compression process for each client and (2) the balanced fine-tuning process with the compressed dataset for the global model in the server. For each client, the compressed dataset is trained for K iterations, and in each iteration, as mentioned earlier; the model θ is randomly initialized from the distribution P_θ. Note that this does not affect the performance of federated learning because θ is only used during the data compression process. During one iteration, the mini-batch $B_{\mathcal{D}}^{c}$ is sampled only from the same class c of the local data of each client, and the loss is then calculated using Eq. (6) with respect to the condensed dataset S_i^c of the same class c.

Next, after clients send their local model along with the compressed form of local data to the server, the server aggregates the client models and then fine-tunes the aggregated global model with the condensed and balanced dataset as follows:

$$\tilde{\theta}^t = \theta^t - \eta \nabla \ell_s(\theta^t) \tag{7}$$

$$\text{such that } \ell_s = \frac{1}{|S|}\sum_{s\in S} \ell_{ce}(\theta^t; s),$$

where $S = \bigcup_{i=1}^{M} S_i$ is the condensed dataset collected from the clients. Note that the condensed datasets generated in the previous rounds are no longer used. Furthermore, in federated learning, clients often need an efficient and fast training method because it is important for each client to quickly learn local data with limited computing resources. The existing dataset compression methods, however, train θ for a certain number of iterations [15] in the middle of matching gradients. In our experiments, it is confirmed that this supplement training of θ does not make significant difference in terms of the quality of the resulting condensed data. Thus, we observe that how well θ is trained is not essential for the task of dataset condensation, but it suffices to vary the value of θ across epochs just like the random initialization of models. To speed up the dataset condensation process, we discard this supplement training process of θ when each client compresses local data.

Moreover, we increase the learning rate (e.g., $\eta = 0.1$ to $\eta = 3.0$) to further improve the training efficiency of dataset condensation. Because this high learning rate can cause another training instability, we additionally perform gradient clipping in the optimization process as follows:

$$g \leftarrow \frac{gv}{\|g\|} \; if \; \|g\| > v\,. \tag{8}$$

Gradient clipping effectively prevents the gradient vanishing problem by normalization when the norm of the gradient g exceeds a certain threshold.

D. FedDC: Stable and Privacy-Protecting Federated Learning

It is important to recall that the goal of our dataset compression for federated learning is not generating synthetic data that are visually similar to real data, but rather disclosing the visual appearance of real data while effectively summarizing important features for fine-tuning. This is different from the normal dataset compression method in the sense that the more visually realistic the compressed resulting images, the more effective the method shows. Thus, there is a trade-off between the quality of compressed data and the degree of privacy protection. The better we train the synthesized data to compress real data, the more difficult it is to protect privacy information.

In order to address this trade-off issue, we suggest using momentum and weight regularization during training of the synthesized data. This way, the condensed image is transformed into one with noise so that it is not visually distinguishable. To be shown in our experiments, the algorithm using this noisy dataset, which we call FedDC+, still maintains a similar performance when compared to the algorithm using the normally compressed dataset. Thus, FedDC+ can achieve both training stability and privacy protection.

V. EXPERIEMENTS

A. Experimental Setup

We conduct our experiments using the CIFAR-10 dataset [26]. CIFAR-10 consists of 50,000 training images and 10,000 images of 10 different classes, and the sizes of all the images is 32×32. To simulate a non-IID scenario, local data of the clients are assigned using the Dirichlet distribution [11, 13]. More specifically, the ratio $\boldsymbol{p_{c,m}}$ of the training data of class $\boldsymbol{c}$ is sampled from $\boldsymbol{p_c \sim Dir_M(\alpha)}$ and assigned to the $\boldsymbol{m}^{\text{th}}$ client. If the concentration parameter $\boldsymbol{\alpha}$ of the Dirichlet distribution is small, the client may have very little data or even no data for some classes. Thus, the smaller the α, the more biased the data distribution of the client is. In this way, we can simulate a non-IID environment, where clients have a biased class ratio, and hence have different quantities of training data.

Fig. 3 displays the number of classes per client and the amount of data per class. We set $\boldsymbol{\alpha}$ to three different values, namely 0.05, 0.1, and 1, each of which considers a different level of non-IID scenario. Most of the existing works deal with either the non-IID environment, where

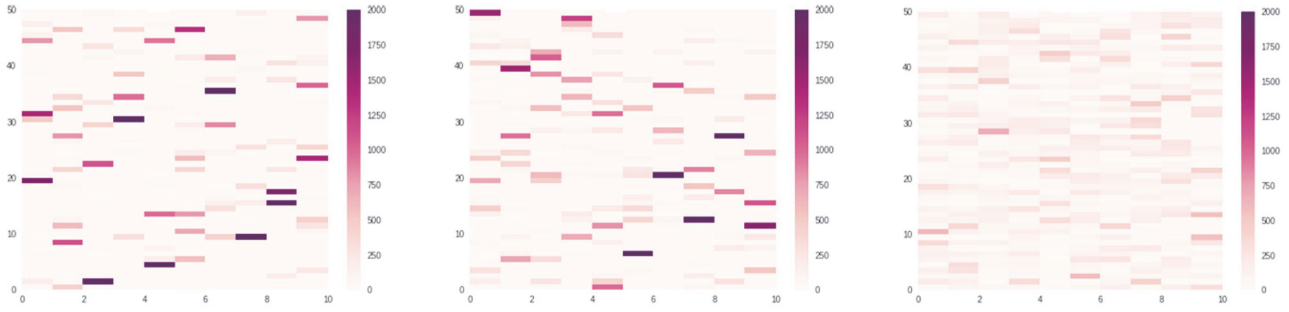

Fig. 3. Label distribution of CIFAR-10 across clients in different data partitions: (a) α = 0.05, (b) α = 0.1, and (c) α = 1.

the amount of local data are all equal across clients even if the number of classes are different, or a weak non-IID scenario (e.g., $\alpha = 0.1$) where the local data volume of a client can slightly be different from those of the other clients. In this paper, we also test a more extreme non-IID case with $\alpha = 0.05$ in order to consider many real-world applications, where 80% of the data often originate from only a small number of clients, approximately 20%, according to Pareto's law.

For the neural network architectures, we use a three-layered convolutional neural network (CNN), which is a model frequently adopted by a typical federated learning experiment [10-12], and also VGG-11 is tested to see whether our FedDC method works well in a deeper architecture.

FedDC is compared to local regularization-based federated learning methods, all of which aim to resolve data heterogeneity, namely FedAvg [1], FedProx [8], and SCAFFOLD [9]. All the algorithms were implemented in PyTorch and trained on a machine with NVIDIA Quadro RTX 6000 and Intel Core Xeon Gold5122. In all the federated learning scenarios, 10 were randomly selected out of 50 clients to participate in each round, and the algorithms run for 500 communication rounds in total with local training epochs of 5 and 10, the learning rate of 0.01, the weight decay of 10^{-3}, and using the SGD optimizer. To create the condensed dataset, 500 iterations are performed with the SGD optimizer and the learning rate of 3.0, to generate one condensed image per class of clients. When fine-tuning with synthetic data on the server, we proceed 10 epochs with the learning rate of 0.01, which is the same as the client's learning rate. The momentum for FedDC+ is 0.9, and the weight decay is 4×10^{-5}.

B. Experimental Results

In Fig. 4, it is well observed that FedDC shows the most stable performance across rounds, compared to the other methods. To precisely quantify the training stability, we also measure the (largest) average accuracy drop and the average accuracy increase over the rounds, which together indicate *round variance*. As shown in Table 1, FedDC and FedDC+ always show the smallest round variance. For instance, when alpha is 0.05 and E is 1 in Table 1, the maximum accuracy drop is around 21% for both FedAvg and FedProx, but FedDC only shows at most a 5.9% accuracy drop. Also, in the same setting, the

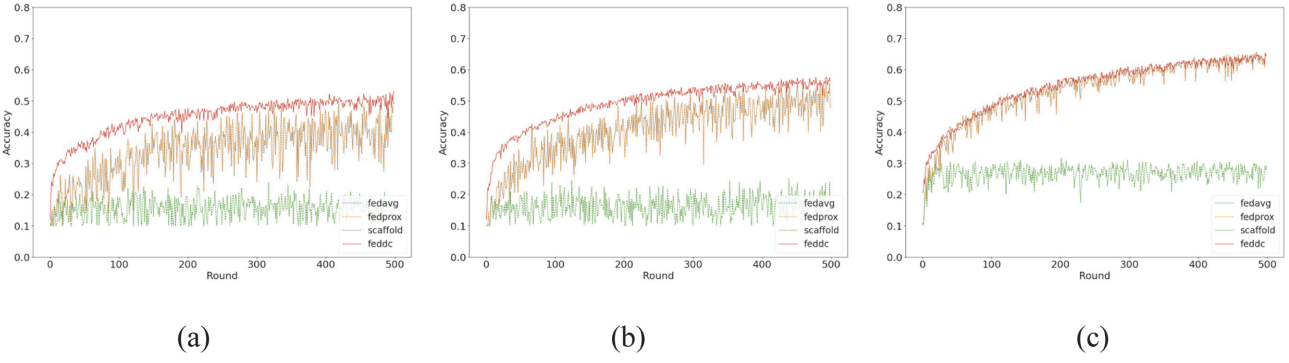

Fig. 4. Comparison of the accuracy for each non-IID on CIFAR-10 using E = 1: (a) α = 0.05, (b) α = 0.1, and (c) α = 1.

Table 1. Comparison of accuracy, average class variance, and average round variance on CIFAR-10

Local epochs	Non-IID	Method	Top-1 accuracy (%)		Class variance	(Max accuracy drop) Avg. accuracy drop / Avg. accuracy increase
			100 rounds	500 rounds		
E = 10	0.05	FedAvg	46.09	55,91	29.17	(-22.64) -6.22 / 5.74
		FedProx	47.11	56.31	27.47	(-18.82) -5.13 /4.97
		SCAFFOLD	33.71	38.04	33.26	(-16.49) -5.65 / 5.08
		FedDC (ours)	**52.72**	**56.63**	**18.61**	(-9.04) -2.11 / 2.30
		FedDC+ (ours)	51.12	56.35	18.67	**(-8.22) -2.02 / 1.91**
	0.1	FedAvg	57,17	61.38	23.82	(-21.05) -4.25 / 4.13
		FedProx	56.77	61.55	22.54	(-18.98) -3.76 / 3.54
		SCAFFOLD	40.10	45.4	31.75	(-17.38) -4.77 / 4.64
		FedDC (ours)	**58,34**	61.65	17.04	(-7.47) -1.88 / 1.78
		FedDC+ (ours)	57.80	**61.80**	**16.74**	**(-6.79) -1.68 / 1.57**
	1	FedAvg	**65.59**	67.37	15.07	(-8.78) -1.76 / 1.92
		FedProx	64.76	67.54	13.13	(10.46) -1.23 / 1.25
		SCAFFOLD	50.79	51.57	20.42	(-11.58) -1.90 / 1.95
		FedDC (ours)	64.76	**67.78**	13.12	**(-5.46)** -1.24 / 1.25
		FedDC+ (ours)	64.24	67.10	**13.03**	(-7.25) **-1.02 / 1.06**
E = 1	0.05	FedAvg	34.89	49.9	30.74	(-21.71) -5.59 / 5.65
		FedProx	34.89c	50.02	30.58	(-21.23) -5.40 / 5.60
		SCAFFOLD	20.86	23.85	31.14	(-11.76) -3.24 / 3.26
		FedDC (ours)	**42.73**	**53.06**	**15.36**	**(-5.9)** -1.36 / 1.35
		FedDC+ (ours)	41.44	52.17	16.08	(-6.32) **-1.05 / 1.16**
	0.1	FedAvg	42.62	56.54	27.04	(-22.06) -4.89 / 4.48
		FedProx	42.63	56.61	26.92	(-21.89) -4.75 / 4.53
		SCAFFOLD	23.83	25.28	29.69	(-13.53) -3.5 / 3.67
		FedDC (ours)	**44.43**	**57.60**	**15.34**	(-3.65) -1.03 / 1.08
		FedDC+ (ours)	43.81	57.50	15.61	**(-3.38) -0.97 / 1.00**
	1	FedAvg	**48.52**	**65.67**	**18.12**	(-9.59) -1.96 / 2.18
		FedProx	48.52	65.62	18.10	(-9.41) -1.95 / 2.16
		SCAFFOLD	30.57	31.78	24.47	(-8.0) -2.20 / 2.35
		FedDC (ours)	48.49	65.06	12.72	(-3.53) -0.83 / 0.91 -
		FedDC+ (ours)	47.86	64.09	**12.62**	**(-3.51) -0.61 / 0.70**

The best results are highlighted in bold.

average accuracy of both FedAvg and FedProx decrease to 5.59% and 5.40%, respectively, but FedDC and FedDC+ maintain stable performance with an average accuracy drop of about 1%. Thus, the maximum drop in accuracy of FedDC is almost as small as the average accuracy drop of the compared methods. SCAFFOLD is somewhat more stable than FedAvg and FedProx, but its learning curve is much worse than the other algorithms due to its strong constraints on training.

FedDC not only shows the most stable performance, but also achieves the best accuracy in most of the rounds. The earlier the rounds and the more extremely non-IID the data, the better FedDC works than the other algorithms. For instance, when $\alpha = 0.05$ (i.e., highly non-IID) for 100 rounds (i.e., earlier rounds), FedDC shows about 7% higher best accuracy than the other algorithms for both E = 1 and E = 10, exhibiting a faster convergence speed as well as the superior performance. The fastest convergence

of our FedDC method is highly beneficial for federated learning so that we can reduce the communication cost by early stopping the training process.

Over the 500 rounds, even if FedAvg and FedProx occasionally achieved a higher accuracy than FedDC in a particular round, they cannot maintain such accuracy even in the next round due to their training instability. Many times, it is observed that they show 10%–20% accuracy drop in the next round after achieving the best performance. In other words, we cannot trust the performance of these existing algorithms in practice in the sense that their performance can decrease by 20% in the worst case of a strong non-IID scenario. Meanwhile, FedDC is managed to achieve accuracy within 1% of the best performance for all 500 rounds, regardless of the data distribution of clients.

In Fig. 5, the overall performance and class variance of FedAvg and FedDC are presented. For each round, class variance is defined as the average of the variances of all the classes, which measures how dispersed the performance of each class is compared to the average performance. Note that the class variance in Table 1 is the average of the class variances of all rounds, whereby the changes in the class variance of the model being trained by each method can be observed. Likewise, FedDC shows not only the lowest round variance (i.e., overall performance change of the round) but also the lowest class variance (i.e., performance difference for each class). By contrast, other algorithms maintain neither constant performance between classes nor constant overall performance within the round. As described in Section III, this is because there are classes with extremely low performance in federated learning due to the round-wise class imbalance. Therefore, even if the global model has a good average performance, some clients can perform extremely poorly if they have only classes with low performance. Meanwhile, FedDC achieves balanced as well as stable performance across classes.

Unlike centralized learning, existing federated learning algorithms cannot take advantage of deeper architectures, and hence they show similar or even worse performance when the model gets deeper [27]. This is also observed in our experiments as FedAvg and FedProx do not show much difference in accuracy for VGG and CNN, both exhibiting an accuracy of approximately 50%. However, FedDC achieves 57% accuracy on VGG under the same highly non-IID situation with $\alpha = 0.05$, which is higher than that of the three-layer CNN with 53% accuracy. Thus, FedDC is not only more stable but also shows even higher performance when the deep architecture VGG is used, as shown in Fig. 6.

Fig. 7 presents a condensed image generated from the real data of the 5th client. The 5th client has 33 airplane

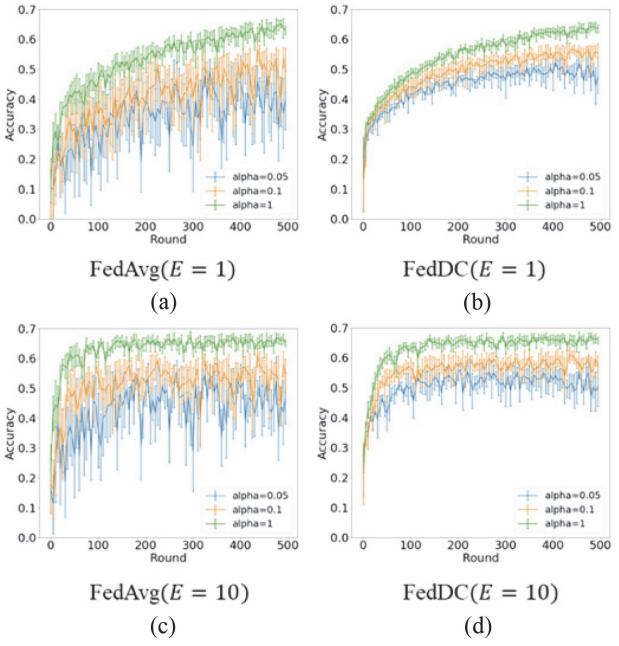

Fig. 5. Comparison of the round-wise class variance: (a) FedAvg (E = 1), (b) FedDC (E = 1), (c) FedAvg (E = 10), and (b) FedDC (E = 10).

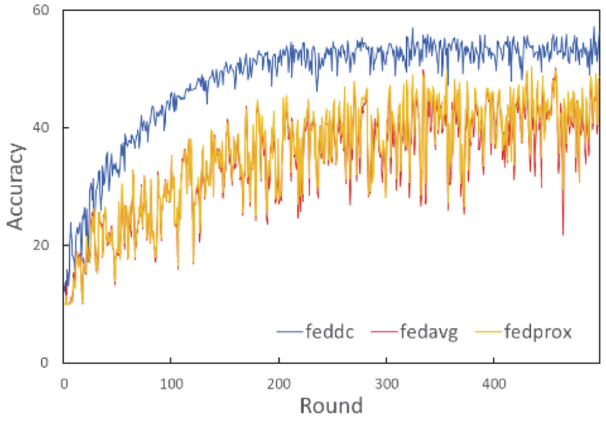

Fig. 6. Comparison of the accuracy in VGG.

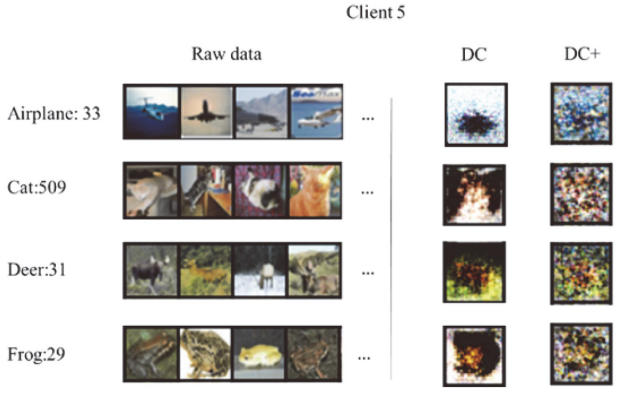

Fig. 7. Visualization of raw data and condensed images by DC and DC+ of the 5th client.

images, 509 cat images, 31 deer images, and 29 frog images. It is impossible to visually distinguish to which class each condensed image of DC belongs even though the condensed image is still effective for making the global model more stable. In the case of DC+, as explained in Section IV, by adding additional momentum and weight attenuation, it is possible to obtain a condensed image with noise, which is even more difficult to recognize its class by human eyes. However, the actual performance is not much different from that of FedDC in Table 1, and even there are cases where the round variance of FedDC+ is lower than FedDC.

VI. CONCLUSION

In this paper, a new federated learning method, FedDC, was proposed to solve the temporary class imbalance problem with a non-IID dataset by dataset condensation. Through various experiments, it was demonstrated that FedDC achieves stable yet balanced performance as well as fast convergence. Based on the proposed method, it is expected that a stable model can always be provided in a federated learning application even when the intermediate performance of the model cannot be validated.

CONFLICT-OF-INTEREST STATEMENT

The authors declare that there is no conflict of interest.

ACKNOWLEDGMENTS

This research was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2021R1F1A1060160) and in part by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-01389, Artificial Intelligence Convergence Research Center of Inha University).

REFERENCES

1. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, 2017, pp. 1273–1282.
2. M. Aledhari, R. Razzak, R. M. Parizi and F. Saeed, "Federated learning: a survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699-140725, 2020.
3. J. Wang, Z. Charles, Z. Xu, G. Joshi, H.B. McMahan, M. Al-Shedivat, et al., "A field guide to federated optimization," 2017 [Online]. Available: https://arxiv.org/abs/2107.06917.
4. A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2018 [Online]. Available: https://arxiv.org/abs/1811.03604.
5. W. Li, F. Milletari, D. Xu, N. Rieke, J. Hancox, W., Zhu, et al., "Privacy-preserving federated brain tumour segmentation," 2019 [Online]. Available: https://arxiv.org/abs/1910.00962.
6. I. Dayan, H. R. Roth, A.Zhong, A. Harouni, A. Gentili, A. Z. Abidin, et al., "Federated learning for predicting clinical outcomes in patients with COVID-19," *Nature Medicine*, vol. 27, no. 10, pp. 1735-1743, 2021.
7. X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-iid data," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.
8. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems (MLSys)*, Austin, TX, 2020, pp. 429-450.
9. S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Virtual Event, 2020, pp. 5132-5143.
10. Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Virtual Event, 2021, pp. 10713-10722.
11. J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," 2020 [Online]. Available: https://arxiv.org/abs/2007.07481.
12. S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konecny, S. Kumar, and H. Brendan McMahan, "Adaptive federated optimization," 2020 [Online]. Available: https://arxiv.org/abs/2003.00295.
13. H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020 [Online]. Available: https://arxiv.org/abs/2002.06440.
14. T. Wang, J. Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," 2018 [Online]. Available: https://arxiv.org/abs/1811.10959.
15. B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," 2020 [Online]. Available: https://arxiv.org/abs/2006.05929.
16. B. Zhao and H. Bilen, "Dataset condensation with differentiable Siamese augmentation," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, Virtual Event, 2021, pp. 12674-12685.
17. Y. X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," *Advances in Neural Information Processing Systems*, vol. 30, pp. 7029-7039, 2017.
18. H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-1284, 2009.
19. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol.

16, pp. 321-357, 2002.
20. C. Drummond and R. C. Holte, "C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling," in *Proceedings of the Workshop on Learning from Imbalanced Datasets II*, Washington, DC, 2003, pp. 1-8.
21. T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 2999-3007.
22. B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.
23. L. Wang, S. Xu, X. Wang, and Q. Zhu, "Addressing class imbalance in federated learning," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, Virtual Event, 2021, pp. 10165-10173.
24. T. M. H. Hsu, H. Qi, and M. Brown, "Measuring the Effects of non-identical data distribution for federated visual classification," 2019 [Online]. Available: https://arxiv.org/abs/1909.06335.
25. I. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.
26. A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Canada, *Technical Report*, 2009.
27. H. Y. Chen and W. L. Chao, "FedBE: making Bayesian model ensemble applicable to federated learning," 2020 [Online]. Available: https://arxiv.org/abs/2009.01974.

Seong-Woong Kim

Seong-Woong Kim received the B.S. degree in computer science from Inha University, South Korea, in 2020. He is currently pursuing the M.S. degree in artificial intelligence at Inha University. His research interests include scalable machine learning and federated learning.

Dong-Wan Choi https://orcid.org/0000-0003-3122-7518

Dong-Wan Choi received the B.S. degree from Inha University, South Korea, in 2008, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2010 and 2014, respectively. From 2016 to 2017, he worked as a Research Associate with the Department of Computing, Imperial College London, U.K., and a Postdoctoral Fellow with the School of Computing Science, Simon Fraser University, Canada, from 2015 to 2016. He is currently an Associate Professor with the Department of Computer Engineering, Inha University. He is working on designing more efficient and continual deep learning algorithms. His research interests include the areas of data mining, databases, and scalable machine learning.