Federated Continuous Learning With Broad Network Architecture

Junqing Le, Xinyu Lei, *Member, IEEE*, Nankun Mu, *Member, IEEE*, Hengrun Zhang, Kai Zeng, *Member, IEEE*, and Xiaofeng Liao, *Fellow, IEEE*

***Abstract*—Federated learning (FL) is a machine-learning setting, where multiple clients collaboratively train a model under the coordination of a central server. The clients' raw data are locally stored, and each client only uploads the trained weight to the server, which can mitigate the privacy risks from the centralized machine learning. However, most of the existing FL models focus on one-time learning without consideration for continuous learning. Continuous learning supports learning from streaming data continuously, so it can adapt to environmental changes and provide better real-time performance. In this article, we present a federated continuous learning scheme based on broad learning (FCL-BL) to support efficient and accurate federated continuous learning (FCL). In FCL-BL, we propose a weighted processing strategy to solve the catastrophic forgetting problem, so FCL-BL can handle continuous learning. Then, we develop a local-independent training solution to support fast and accurate training in FCL-BL. The proposed solution enables us to avoid using a time-consuming synchronous approach while addressing the inaccurate-training issue rooted in the previous asynchronous approach. Moreover, we introduce a batch-asynchronous approach and broad learning (BL) technique to guarantee the high efficiency of FCL-BL. Specifically, the batch-asynchronous approach reduces the number of client–server interaction rounds, and the BL technique supports incremental learning without retraining when learning newly produced data. Finally, theoretical analysis and experimental results further illustrate that FCL-BL is superior to the existing FL schemes in terms of efficiency and accuracy in FCL.**

***Index Terms*—Broad learning (BL), catastrophic forgetting, continuous learning, federated learning (FL).**

Manuscript received January 14, 2021; revised June 8, 2021; accepted June 13, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61932006, Grant 61772434, and Grant 61806169; in part by the National Key R&D Program of China under Grant 2018AAA0100101; in part by the China Postdoctoral Science Foundation under Grant 2018M643085; and in part by the scholarship from China Scholarship Council (CSC). This article was recommended by Associate Editor Y.-M. Cheung. *(Corresponding author: Xiaofeng Liao.)*

Junqing Le is with the College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China (e-mail: lejunqing@163.com).

Xinyu Lei is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: leixinyu@msu.edu).

Nankun Mu and Xiaofeng Liao are with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, and College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: nankun.mu@qq.com; xfliao@cqu.edu.cn).

Hengrun Zhang and Kai Zeng are with the Department of Computer Science and the Department of Electrical and Computer Engineering, respectively, George Mason University, Fairfax, VA 22030 USA (e-mail: hzhang18@gmu.edu; kzeng2@gmu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2021.3090260.

Digital Object Identifier 10.1109/TCYB.2021.3090260

I. Introduction

VARIOUS end devices (e.g., smartphones) produce a wealth of data, which can be used to train learning models for different purposes (e.g., text classification [1], pattern recognition [2]). To train a learning model, the traditional approach requires all devices (i.e., clients) to submit their local training data to a cloud server. However, outsourcing the client's local training data to a central server (e.g., cloud server) may lead to serious security issues because the client's local training data are visible and controlled by the central server. The client's local training data may be highly security sensitive, so the client does not want to share them with other parties. For example, the training data may contain financial records, healthcare records, location information, etc. To address this issue, the federated learning (FL) technique is proposed to enable multiple clients to jointly train a shared global learning model while keeping its own local training data hidden from the central cloud server. In one round of training, each client computes its local weight and uploads it to the central cloud server. Then, the server helps for the aggregation of locally computed weights (i.e., computing the weighted average of locally computed weights to obtain the global weight). A global model can be trained after numerous rounds of client–server interactions. Because the raw training data are not required to upload to the cloud server (only the locally computed weights are uploaded), the privacy of training data is protected.

Currently, most of the proposed FL models focus on one-time learning without consideration for continuous learning. Continuous learning (or lifelong learning) describes a learning process in which a model is continuously trained on a stream of training data. Based on past knowledge, continuous learning can learn new knowledge and summarize it in the new learning model. Consequently, the continuously updated learning model can adapt to environmental changes and provide better real-time performance in practical applications. Therefore, it is imperative to develop techniques for federated continuous learning (FCL). However, the existing FL techniques cannot be directly applied or simply extended to enable FCL. First, the previous FL models lacked the continuous learning capability, so they suffered from the *catastrophic forgetting problem* [3]. In a dynamic scenario, where the clients

can join and leave the network randomly, the newly received weights can be significantly different from the previous ones. Thus, the new aggregated learning model may "forget" the knowledge learned previously. An FCL scheme being able to tackle the catastrophic forgetting problem is desirable but missing in the literature. Second, there are several limitations in the existing global weight update approaches adopted in FL. The existing global weight update approaches mainly have two types: 1) asynchronous and 2) synchronous. In a synchronous approach, the time consumption is determined by the client of the slowest uploading among all of the clients during aggregating, so it may cause a serious slow-training issue. To avoid the slow-training issue, we can resort to the asynchronous approach in which the clients upload their updates to the cloud server without waiting for the slowest client. However, the existing asynchronous approach has two issues: 1) inaccurate-training issue and 2) communication-inefficient issue.

1) *Inaccurate-Training Issue:* There is the possibility of stale updates in an asynchronous approach [4], [5]. The stale update of one client could overwrite the updates of other clients in the aggregation process, resulting in an inaccurate-training issue.
2) *Communication-Inefficient Issue:* If an asynchronous approach is adopted, then the required number of client–server interaction round is growing rapidly with an increasing number of clients. How to reduce the client–server interaction round should be considered in FCL.

In this article, we propose an FCL scheme based on broad learning (FCL-BL) scheme to support efficient and accurate FCL. FCL-BL is developed based on the following three key techniques.

1) To address the catastrophic forgetting problem, we develop a *weighted processing strategy*. On receipt of the global weight from the cloud server, the weighted processing strategy lets each client compute the weighted average of the global weight (i.e., received weight from the server) and the client's local weight. The weighted average serves as the client's next update. Based on the update, the newly trained global model can preserve the previously learned knowledge and also adapt to the new knowledge. Thus, the catastrophic forgetting problem is solved.
2) To tackle the inaccurate-training issue, we design a *local-independent training solution*. In FCL-BL, the local training process can be performed independently (the local training model does not rely on the knowledge of the global learning model). Once the new local weight is obtained, the client can always use the latest global weight to compute the new update that is uploaded to the cloud server. Therefore, the possibility of stale updates is eliminated and the inaccurate-training issue is tackled.
3) To solve the communication-inefficient issue, we customize the exiting asynchronous global weight update approach (called the *batch-asynchronous approach*). In the batch-asynchronous approach, the cloud server computes the global weight only after it receives a batch (i.e., subset) of clients' updates. By properly adjusting the cardinality of the subset, the batch-asynchronous approach can greatly reduce the number of client–server interaction rounds while ensuring efficient aggregation (i.e., the high prediction accuracy of the global model and short aggregation time).

The major technical challenge in the FCL-BL design is how to achieve continuous learning efficiently for each client. To preserve the gained historical knowledge, the client can use all of the historic datasets and the newly produced dataset in learning. However, this approach is not computationally efficient because it requires each client to retrain the model from scratch. To address this challenge, we take advantage of broad learning (BL). BL supports effective and efficient incremental learning without the need for the deep architecture [6]. In BL, the network expands neurons broadly, and only the parameters connected to the output are adjusted by linear regression, unlike deep learning, which tunes abundant connecting parameters in filters and layers. In addition, the BL updates parameters incrementally when new data enter the network, which means BL can directly add input data to the original model without reconstructing. The process of parameter tuning and remodeling is simplified greatly in the BL. Therefore, the training process of FCL-BL is significantly accelerated. Moreover, the simple network architecture of BL can help to reduce the communication volume for FL, and the incremental learning of BL is suitable for learning streaming datasets.

To sum up, the main contributions of this article are described as follows.

1) We propose the FCL-BL scheme to support continuous learning in the FL setting. To solve the catastrophic forgetting problem, we propose a weighted processing strategy. The FCL-BL can preserve the existing learned knowledge and also adapt to the new knowledge, providing better real-time performance in practice.
2) We develop a local-independent training solution used in the customized asynchronous global weight update approach. The proposed solution enables us to avoid using a time-consuming synchronous approach while addressing the inaccurate-training issue rooted in the existing asynchronous approach. Therefore, FCL-BL achieves both fast and accurate training.
3) We introduce a batch-asynchronous approach and a BL technique to guarantee the high efficiency of FCL-BL. On the one hand, the batch-asynchronous approach can greatly reduce the number of client–server interaction rounds while largely preserving the prediction accuracy of the global model and avoiding long aggregation delay. On the other hand, the BL technique supports incremental learning without requiring each client to retrain when learning newly produced data.
4) The experimental results demonstrate that FCL-BL can ensure the high prediction accuracy of the global trained model even if each client owns an imbalanced and nonindependent and identically distributed (Non-IID) dataset.

The remainder of this article is organized as follows. In Section II, some related works are reviewed. The preliminaries about FCL-BL are introduced in Section III. Section IV describes the overview of FCL-BL, including the system model, assumptions, and goals. Section V presents the design of a global weight update in FCL-BL. In Section VI, FCL-BL is constructed to support continuous learning, and the related algorithms are described in detail. The analysis of FCL-BL is presented in Section VII. Performance evaluations are conducted in Section VIII, followed by the conclusions and the future work in Section IX.

II. Related Work

FL is an efficient and secure scheme for the distributed network and focuses on training heterogeneous datasets. Konečnỳ *et al.* [7] first introduced the federated optimization for the distributed data. Then, McMahan *et al.* [8] proposed communication-efficient learning, called FedAvg, which reduces the communication cost in update transmission and can train Non-IID data. After that, a variety of techniques [9]–[12] had been proposed to make the FL more computationally efficient and accurate. The schemes in [13] and [14] proposed the uplink communication-efficient approaches to featurewise distributed data. In the realistic wireless networks, the methods based on user selection [15] and resource optimization [16], [17] were used to improve the communication of FL. However, these schemes suffer from the communication-inefficiency issue caused by the inherent constraints of a synchronous approach.

Different from the above synchronous schemes, some asynchronous FL schemes have been proposed in [18] and [19], but they have the inaccurate-training issue due to stale updates. In addition, the above synchronous and asynchronous FL schemes do not support continuous learning. The asynchronous online FL scheme in [20] focuses on the continuous streaming data, but the catastrophic forgetting problem is not addressed.

To solve the catastrophic forgetting problem in continuous learning, some feasible schemes are proposed in [21]–[23]. Specifically, in the work of learning without forgetting (LwF) [21], the new data are inputted into the old network to generate virtual data, and all virtual data are used to recall old knowledge. The objective function of LwF contains the constraints of the old network and trains the new task on the entire network every time. Kirkpatrick *et al.* [22] proposed elastic weight consolidation (EWC) to adjust weights (network parameters) according to the importance of weights in the old tasks and then train new tasks on these weights. However, these works do not consider the distributed system and cannot be used to prevent the catastrophic forgetting problem caused by dynamic clients. The work (Fed-ADP) in [23] focuses on continuous learning and FL. It aims to continuously train with multiple clients on a sequence of tasks from the private local data stream. Unfortunately, it ignores the training of a global model and still cannot solve the catastrophic forgetting of the dynamic clients.

TABLE I
Comparison Between FCL-BL and Existing FL Schemes
(●: Yes, ○: No, ◐: Partly)

Performance \ Schemes	[7–17]	[18, 19]	[20]	[21–23]	FCL-BL
Support distributed system	●	●	●	○	●
Support continuous learning	○	○	●	●	●
No catastrophic forgetting	○	○	○	◐	●
No communication-inefficiency issue	○	●	●	○	●
No training-inaccuracy issue	●	○	○	●	●

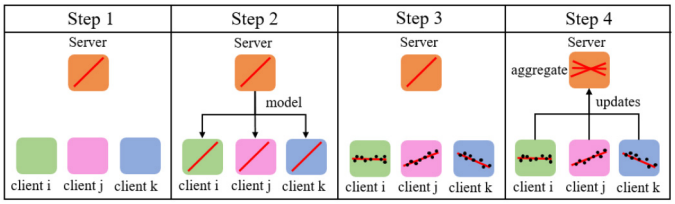


Fig. 1. Four steps of an FL round.

Different from the horizontal FL considered in this article, the vertical FL focuses on the cases, where all participating parties share the same sample space but differ in the feature space. Some related works about vertical FL are presented in [24] and [25]. Moreover, some continuous learning schemes [26], [27] have been proposed to learn the multidomain data. These are interested topics but beyond the scope of this article.

Compared to the above schemes [7]–[23], the proposed FCL-BL shows great advantages. The comparison results are summarized in Table I.

III. Preliminaries

In this section, we introduce some related preliminaries and mathematical notations.

FL: FL is a machine-learning setting, which multiple decentralized clients collaboratively train a model under the orchestration of a central server, and each client's raw data are not exchanged and transferred [28]. In the FL setting, each client independently computes weights based on its local data and uploads the trained weights to a server, where the client-side weights are aggregated to obtain a global model. Each client–server interaction for a global model calculating is defined as an FL round. In general, the processes of FL can be decoupled into multiple rounds, and each round consists of four steps which are shown in Fig. 1.

1) The server chooses a statistical model to be trained.
2) The server sends the initial model or the global weight to clients.
3) Clients train the received model independently with their data and obtain local weights, which are uploaded to the server.
4) The server updates the global weight by computing the weighted average of the received local weights.

Assume that there are N clients and the set of indices of samples is $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$, where the set u_i stores the indices of samples in client i. Let (x_k, y_k) be a sample of client i. The samples of client i can be denoted as $(X_i, Y_i) = \{(x_k, y_k)\}_{k \in u_i}$. The sample number of client i is represented as $n_i = |u_i|$. The loss function on the sample (x_k, y_k) with model parameter (i.e., model weight) ω is typically defined as $f_k(\omega) = \ell(\omega; x_k, y_k)$. In the FL, the goal of model training is to find a weight ω that minimizes the loss. The minimization of the loss function can be expressed as $\min_\omega f(\omega) = \sum_{i=1}^{N} (n_i/n) F_i(\omega)$, where $F_i(\omega) = (1/n_i) \sum_{k \in u_i} f_k(\omega)$ and $n = |\mathcal{U}|$ represents the number of total samples in training.

Federated stochastic gradient descent (FedSGD) [29] is the direct transposition of stochastic gradient descent (SGD) to the federated setting, where the gradients are calculated on all local datasets of a client. Then, the gradients are averaged by the server and used to make a gradient descent step.

Federated averaging (or FedAvg) [8] is a generalization of FedSGD, which exchanges the updated weights rather than the gradients. In a typical implementation of FedAvg, it randomly selects a part of clients for the aggregation in each round. Each elected client computes $g_i = \nabla F_i(\omega_t)$ with a fixed learning rate η based on its local data, where t represents the index of the FL round. Specifically, the local update $\omega^i \leftarrow \omega^i - \eta g_i$ is iterated multiple times before the aggregation step in FedAvg. Then, the calculated update ω_{t+1}^i is taken as the weight of client i in the next round and uploaded to the server. Finally, according to the aggregation equation $\nabla f(\omega_t) = \sum_{i=1}^{N} (n_i/n) g_i$, the global weight in the server can be written as $\omega_{t+1} \leftarrow \sum_{i=1}^{N} (n_i/n) \omega_{t+1}^i$, where ω_{t+1} is the global weight at $t+1$ round. Currently, FedAvg as an efficient and robust technique has been used widely in the learning of distributed data.

Continuous Learning: Continuous learning (or lifelong learning) is proposed by Thrun [30], which is a long-studied topic with a vast literature. It can learn continuously from streaming data, building a model on what was learned previously, while being able to reapply, adapt, and generalize it to new situations. In this part, we only discuss some recent relevant works that may be applied in the distributed network.

Fed-Regularizer: Adding a regularized term (the ℓ_1-norm and ℓ_2-norm) into the loss function is a simple method to alleviate catastrophic forgetting and achieve continuous learning. For the FL, the loss function with the ℓ_2-norm can be rewritten as

$$\min_{\omega_t} f(\omega_t) = \sum_{i=1}^{N} \frac{n_i}{n} F_i(\omega_t) + \lambda \|\omega_t - \omega_{t-1}\|_2$$

where λ is constant coefficient and sets how important the previous model is compared to the new one. If ℓ_1-norm is added, the regularized term represents $\lambda ||\omega_t - \omega_{t-1}||_1$.

Fed-ADP: The continuous learning with additive parameter decomposition (called Fed-ADP) [23] is proposed by Yoon *et al.* It allows to additively decompose parameters to prevent catastrophic forgetting. Assume that there are C clients, and each client c trains a model on a privately accessible sequence of tasks $\mathcal{T}_c^1, \mathcal{T}_c^2, \ldots, \mathcal{T}_c^t \subseteq \mathcal{T}$. The goal of Fed-ADP is to effectively train C continuous learning models on their private task stream. The model parameters ω_c^t for task t at client c can be defined as follows:

$$\omega_c^t = B_c^t \odot m_c^t + A_c^t + \sum_{i \in C} \sum_{j=1,\ldots,t-1} \alpha_{i,j}^t A_i^j$$

where B_c^t is a base parameter for the cth client shared across all tasks, m_c^t is a sparse mask that enables to selectively utilize B_c^t, $\odot$ is an elementwise multiplication, and A_c^t is the highly sparse task-adaptive parameter for the task given at round t. The additively decomposable parameter ω_c^t is learned by optimizing the following objective:

$$\min_{B_c^t, m_c^t, A_c^{\{1:t\}}, \alpha_c^t} \mathcal{L}\left(\omega_c^t, \mathcal{T}_c^t\right) + \lambda_1 \Omega\left(m_c^t, A_c^{\{1:t\}}\right) + \lambda_2 \sum_{i=1}^{t-1} \left\| \Delta B_c^t \odot m_c^i - \Delta A_c^i \right\|_2^2$$

where $\mathcal{L}$ is a loss function, $\Omega(\cdot)$ is a sparsity-inducing regularization term, $\Delta B_c^t = B_c^t - B_c^{t-1}$, and ΔA_c^i is the difference between the task-adaptive parameter for task i at the current and previous round.

BL: BL is a complete paradigm shift in discriminative learning and a very fast and accurate learning technique without deep structure. BL is based on the traditional random vector functional-link neural network (RVFLNN), which was proposed by Chen and Liu [6]. Moreover, BL is incremental learning, which can continuously learn from the new incoming datasets without the entire retraining from the beginning. Currently, BL had been applied in the time-series prediction [31], [32]; control of nonlinear dynamic systems [33]; and image recognition [34].

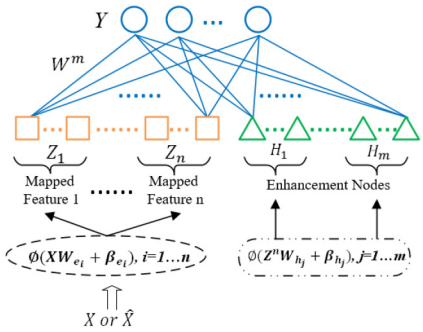


Fig. 2. Framework of BL [6].

The structure of BL, shown in Fig. 2, is established in the form of a flat network. In BL, the original dataset X (or the new dataset $\hat{X}$) is mapped to the feature nodes. The ith feature node Z_i is calculated by function $\phi_i(X\omega_{e_i} + \beta_{e_i})$, where ω_{e_i} is the random weight vector with a proper dimension. ϕ_i and ϕ_k are different functions for $i \neq k$. We denote that $Z^i \equiv [Z_1, \ldots, Z_i]$ is the concatenation of all the first i groups of feature nodes. Then, the structure is expanded in a wide sense via the enhancement nodes. Similarly, the jth group of the enhancement nodes is represented as $H_j = \xi_j(Z^n \omega_{h_j} + \beta_{h_j})$, and the concatenation of the first j groups of the enhancement

nodes is denoted as $H^j \equiv [H_1, \ldots, H_j]$. ξ_j and ξ_r are different functions for $j \neq r$. Hence, the training of the BL model can be represented as $Y = [Z^n|H^m]W$, where $[Z^n|H^m]$ is taken as input and Y is the output. Moreover, W is represented as the model weight, and $|$ is the concatenation operation.

The goal of broad networks is to calculate the connecting weights W^m. According to the above formulas, we can obtain $W^m = [Z^n|H^m]^+Y$, where W^m can be easily computed through the ridge regression approximation of $[Z^n|H^m]^+$ by using the formula $A^+ = \lim_{\lambda \to 0}(\lambda I + AA^T)^{-1}A^T$. In this article, A^+ is represented as the pseudoinverse of A.

BL also supports incremental learning, which can fast update model weights based on the previous networks. The detailed process is shown below. Let the newly produced datasets be $\hat{X}$. Then, we denote $A_n^m = [Z^n|H^m]$, which represents the n groups of feature nodes and m groups of enhancement nodes of the initial network. $\hat{A}_n^m$ represents the updated group of feature nodes and enhancement nodes based on $\hat{X}$, where $\hat{A}_n^m = [\phi(\hat{X}\omega_{e_1} + \beta_{e_1}), \ldots, \phi(\hat{X}\omega_{e_n} + \beta_{e_n})|\xi(\hat{Z}^n\omega_{h_1} + \beta_{h_1}), \ldots, \xi(\hat{Z}^n\omega_{h_m} + \beta_{h_m})]$. Hence, the new weight is calculated by $\hat{W} = W + (\hat{Y}^T - (\hat{A}_n^m)^T W)B$, where $\hat{Y}$ are the corresponding labels of additional $\hat{X}$, $\hat{W}$ is the updated weight, and matrix B satisfies

$$B^T = \begin{cases} (C)^+, & \text{if } C \neq 0 \\ \left(1 + D^T D\right)^{-1}\left(A_n^m\right)^+ D, & \text{if } C = 0 \end{cases}$$

where the superscript T represents the transpose of the matrix, $D^T = (\hat{A}_n^m)^T A_n^{m+}$, and $C = \hat{A}^T - D^T A_n^m$.

Quantity Imbalance: Quantity skew, that is, different clients can hold vastly different amounts of data. For example, clients use the service or application differently, resulting in different amounts of local datasets [8].

Non-IID: This article focuses on the Non-IID caused by label distribution skew (called class imbalance), where the distribution of label (i.e., class) varies across clients. For example, when clients are tied to particular geo-regions, the distribution of labels varies across clients—some animals are only in certain countries or zoos, such as kangaroos in Australia and pandas in China [28]. To achieve a Non-IID setting, the randomly assigned samples for each client contain only a part of labels. Take the MNIST dataset as an example. The training sets are evenly partitioned into 100 clients. For the IID setting, each client is randomly assigned a uniform distribution over ten classes. For the Non-IID setting, the training data are sorted by class and divided to create two extreme cases: one-class Non-IID, where each client receives data partition from only a single class, and two-class Non-IID, where the sorted data are divided into 200 partitions and each client is randomly assigned two partitions from two classes [9].

Mathematical Notations: The mathematical notations used and their semantic meanings are summarized in Table II.

TABLE II
MATHEMATICAL NOTATIONS USED BELOW AND THEIR SEMANTIC MEANINGS

Notations	Semantic meanings
$Data_i$	Data of client i
U_i^j	The total client number of ${}^c\omega_i^j$
$\hat{X}_i^j, \hat{Y}_i^j$	The j-th new input data and output data added in client i, respectively
W_i^m, $\hat{W}_i^m$	m dimension weights of client i, new m dimension weights of client i based on new inputs
t_i^l	The last upload time of client i
gw_m	The global weight of round m
N_A^m, n_i (n_i^j)	The sample number in gw_m and client i (at time j), respectively
ω_i^j, ${}^c\omega_i^j$, $\bar{\omega}_i^j$	The weights that trained on local datasets of client i in local update j; The weights of client i in local update j after local re-aggregation; The updated weights of client i in local update j
$\boldsymbol{W}, \boldsymbol{N}$	The set of the received client's updates and the sample number, respectively
bs, thr_1	The batch size; The sample threshold in clients
$d(\cdot)$	Euclidean distance function
W_{LR}, W_C, W_{M1}	The global weight based on ${}^c\omega_i^j$, ω_i^j and the updates after M1, respectively
W_{global}	The global weight based on the new local weights of all previously received clients and the clients in the aggregation of next round
L, H	The set of the clients in current round of FCL-BL and the next round of FCL-BL, respectively
h_1, h_2, h_3	The set of clients using the methods A_1, A_0 and A_2 of Algorithm 3, respectively

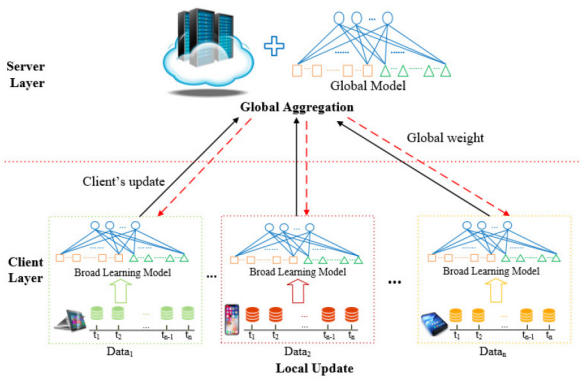


Fig. 3. System model of FCL-BL.

IV. OVERVIEW OF FCL-BL

In this section, we will introduce the system model, assumptions, and goals of FCL-BL.

A. System Model of FCL-BL

As shown in Fig. 3, the system model of FCL-BL includes two layers. *Client layer*, where the clients independently train their own datasets and perform local training based on the BL model, is the source of the data in the network. In each client, the new dataset with *a priori* unknown size can be produced over time in an IID or Non-IID way. *Server layer*, which receives model updates from local clients, performs aggregating and then returns the global weight to clients.

Under the coordination of the cloud server, multiple clients collaboratively train a global model based on their local datasets. Then, the global weight will be updated based on the new datasets in the next round. To solve the catastrophic forgetting problem caused by the newly generated dataset from local clients, the BL model is adopted. Based on the incremental learning of BL for new inputs, it can fast learn new

knowledge without forgetting the previously learned model. Besides, the broad network architecture of FCL-BL helps to mitigate the catastrophic forgetting caused by dynamic clients.

B. Assumptions

We make the following assumptions: 1) the clients are mobile devices, such as smartphones and laptops; 2) the clients can control their data, so they can refuse to upload their updates. The clients may be unstable so they may drop out at any time; 3) the local model can be directly applied to the clients for prediction, classification, etc. These three assumptions are in line with real-world applications to ensure that the proposed scheme can be more scalable; and, in addition, 4) assume that each user uses the same initial model for training.

C. Goals

The proposed FCL-BL aims to achieve the following design goals.

1) *Efficiency:* a) The speed of local training based on BL is faster than that of the deep learning with SGD in FCL-BL. b) The global weight update is communication-efficient and can avoid the slow-training issue.
2) *Accuracy:* a) The prediction accuracy of the global model in FCL-BL should be close to that of a centralized learning model, and higher by comparing with that of the FL model with deep learning; b) FCL-BL should overcome the catastrophic forgetting problem and solve the inaccurate-training issue effectively.

V. FCL-BL With Global Weight Update Design

In this section, we present the design of the global weight update in FCL-BL. We first introduce the proposed local-independent training and then propose a batch-asynchronous approach.

A. Local-Independent Training Solution

The synchronous and asynchronous global weight update approaches are two commonly used methods for updating the global weight. However, they suffer from some limitations as analyzed in the following.

In the synchronous approach, the server updates the global weight when receiving all the client's local weights. Specifically, all clients are trained in parallel, and then they upload their local weights simultaneously when all clients finish local training. However, it can incur a lot of idle time in the fast clients while waiting for the slowest client, meaning the fast clients cannot do anything until the slowest client finishes training. Even worse, the global weight update process may be halted if any client drops out or refuses to upload. Hence, the synchronous approach leads to serious slow training.

In the asynchronous approach, the global weight is updated whenever the server receives a local weight from any client. However, the different training speeds of clients will cause the inaccurate-training issue. For example, if client 1 and client 2 are both trained based on gw_1 and the training speed of client 1 is faster, gw_2 may be updated on ω_1 (i.e., $gw_2 = gw_1 + \omega_1$) and gw_3 may be updated by ω_2 (i.e., $gw_3 = gw_1 + \omega_2$) in the asynchronous approach. When client 3 trains on the latest global weight gw_3 and obtains gw_4 (i.e., $gw_4 = gw_3 + \omega_3$), ω_1 is overwritten by ω_2. If ω_1 provides better training results than ω_2, we take ω_2 as a stale update, which may cause the inaccurate-training issue.

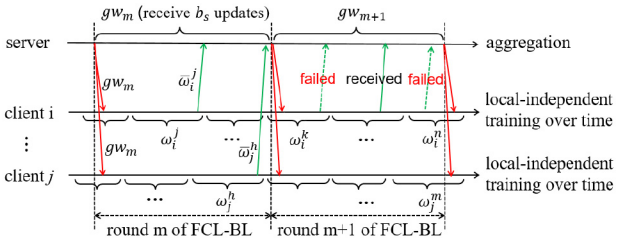


Fig. 4. Batch-asynchronous approach.

To address the slow-training and inaccurate-training issues, we propose a local-independent training solution. In the solution, the local training processing is performed independently. Moreover, to avoid the local training on a too small dataset or large dataset, the sample threshold of the client is set as thr_1, and the local training is performed as long as the sample number is more than thr_1. Thus, the client can train on its new local datasets continuously without waiting for the slow clients anymore, so the slow-training issue of the synchronous approach is tackled. Besides, in local-independent training, each client updates the local weight only based on its own local dataset. That is, the local training model does not rely on the knowledge of the global learning model. Without the constraint of global weight, the clients can always choose the newest global weight to compute the new client's updates.

Hence, the local-independent training solution can guarantee that each global weight update is based on the newest global weight, so the stale update is eliminated and the inaccurate-training issue is solved.

B. Batch-Asynchronous Approach

Although the slow-training issue and inaccurate-training issue can be tackled by the local-independent training solution, there are other limitations in the global weight update. If the local-independent training solution is used in the synchronous approach, the server only performs aggregation to update the global weight when all client's updates are received, so the aggregation time is determined by the slowest client. If the asynchronous approach is adopted, the server performs the global weight update as soon as receiving any client's update, which achieves a short aggregation delay. However, the cost of doing so is to increase the number of interaction rounds to obtain a global model with high prediction accuracy, especially for massive clients, which will lead to a communication-inefficient issue. Toward this end, we propose a batch-asynchronous approach, as shown in Fig. 4.

In the batch-asynchronous approach, the server performs the aggregation only after it receives a batch of client's updates. By properly adjusting the cardinality of the batch (defined as bs), the batch-asynchronous approach can greatly reduce

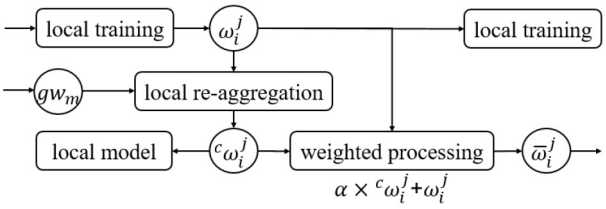


Fig. 5. Local update in clients.

the number of client–server interaction rounds while ensuring efficient aggregation. In addition, to ensure an efficient global weight update and avoid stale updates simultaneously, the client's update can be uploaded at any time. To ensure the fairness of learning, only at most one update of each client can be received by the server in a round of FCL-BL. If the client's update upload fails, the client is allowed to upload again; if the uploading is received by the server, the uploading of this update to the other client will be refused in the same round.

In a round of FCL-BL, there are the following four steps.

1) The server sends the newest global weight gw_m to all clients.
2) Clients perform local training based on their new datasets to obtain the new client's weight ω_i^j independently.
3) Clients calculate their own updates $\bar{\omega}$ based on gw_m, and send $\bar{\omega}$ to server (presented in the next section).
4) The server performs aggregation on the received *bs* client's updates, then computes new global weight gw_{m+1}.

VI. FCL-BL With Continuous Learning

Based on the above design of global weight update, FCL-BL is constructed to achieve continuous learning. In the FCL-BL, the local BL model, the local reaggregation strategy, and the weighted processing strategy are proposed.

In clients, the local update is performed, which is presented in Fig. 5. In a local update, it includes the local training, the local reaggregation strategy, and the weighted processing strategy. To be more specific, in the local training, the new local weight ω_i^j can be computed by using the BL model. Then, in the local reaggregation strategy, the newest global weight gw_m and ω_i^j will be reaggregated to compute ${}^c\omega_i^j$, which can be used to update the local model. Finally, the weighted average of ${}^c\omega_i^j$ and ω_i^j is computed to obtain the new updates $\bar{\omega}_i^j$ in the weighted processing strategy, and $\bar{\omega}_i^j$ will be uploaded to cloud server.

In the server, *bs* client's updates are received and then are aggregated to generate the new global weight gw_{m+1}. Finally, gw_{m+1} will be returned to the clients for the next local update.

The detailed processes of FCL-BL in a round are presented in Algorithm 1, including the local training, the local reaggregation strategy, and the weighted processing strategy.

A. Local Training Based on Broad Learning

Local training based on BL can achieve continuous learning without any catastrophic forgetting for each client. As the dataset of the client is changed over time, the continuous learning in the client may cause catastrophic forgetting. To address this issue, the client can use all of the historic datasets and the newly produced dataset in learning. However, this approach is not computationally efficient, because it requires each client to retrain the local training model from scratch. In this article, we introduce the BL, which employs a broad network architecture for learning and supports incremental learning. In BL, it can continuously learn new incoming datasets without the entire retraining from the beginning, so the training is significantly accelerated. Algorithm 2 shows the local training based on BL in a client. In Algorithm 2, the client produces a new dataset $(\hat{X}_i^j, \hat{Y}_i^j)$. When the sample number of the new dataset is more than thr_1, the local training based on BL is performed, where the new local weight is computed according to (1). Moreover, the total sample number (n_i^j) of the client will also be updated (see step 5). Finally, the new local weight ω_i^j and the sample number n_i^j will be used in the next processing of local update.

Algorithm 1 Proposed FCL-BL

Clients:

Input: $[(\hat{X}_i^j, \hat{Y}_i^j), \ldots, (\hat{X}_k^h, \hat{Y}_k^h)]$ that are the new datasets of all clients, $(\hat{n}_i^j, \ldots, \hat{n}_k^h)$ that is the number of the new samples, ω_i^{j-1} (the current weight), gw_m (the received global weights), **lab** (the label set), $|L|$ (the number of received clients), N_A^m (the total number of samples participating in the aggregation).

Output: The updated weights $[\bar{\omega}_i^j, \ldots, \bar{\omega}_k^h]$ in round m.

1: **Initialization:** The threshold of sample number is thr_1.
2: According to Algorithm 2, the local training is performed to obtain new weight ω_i^j.
3: According to Local Re-aggregation Strategy (Algorithm 3), we can get the local re-aggregated value ${}^c\omega_i^j$.
4: Then calculate $\bar{\omega}_i^j$ based on ${}^c\omega_i^j$ and $[\hat{X}_i^j, \hat{Y}_i^j]$ by Algorithm 3.
5: Upload $[\bar{\omega}_i^j, n_i^j]$ to the server.
6: Each client runs steps 2 to 5, and there is no order between clients.

Server:

Input: $\{(\bar{\omega}_i^j, n_i^j), \ldots, (\bar{\omega}_k^n, n_k^n)\}$ the client's updates with sample number.

Output: $[gw_{m+1}, N_A^{m+1}]$.

1: **Initialization:** The batch size bs.
2: Receive the updates and sample number from clients.
3: **if** $|L| = bs$ **then**
4: Let the set of the received information be represented as $[\boldsymbol{W}, \boldsymbol{N}]$.
5: Calculate the total number of the received samples $N_A^{m+1} = \sum_{num \in \boldsymbol{N}} num$.
6: $gw_{m+1} = \sum_{\omega_l \in \boldsymbol{W}, n_l \in \boldsymbol{N}} (w_l \cdot n_l) / N_A^{m+1}$.
7: **end if**
8: Send the new global weight and its sample number $[gw_{m+1}, N_A^{m+1}]$ to the clients for the next round of learning.

B. Local Reaggregation Strategy

The local reaggregation strategy ensures the local model with high prediction accuracy. In clients, the local model can be applied for prediction, classification, etc. However, as the new datasets are produced over time, if the global weight is directly used to update the local model, the newly produced dataset is not utilized. If the new local weight is used to update the local model, the learned knowledge from other clients is not utilized. Both of them will affect the prediction accuracy of the local model.

Algorithm 2 Local Training in Client i

Input: $(\hat{X}_i^j, \hat{Y}_i^j)$ that are the new samples, $\hat{n}_i^j$ (the number of the new samples), ω_i^{j-1} (the previous weight), n_i^{j-1} (the sample number).
Output: ω_i^j.

1: **if** $\hat{n}_i^j < thr_1$ **then**
2: $\omega_i^j = \omega_i^{j-1}$ and $\hat{n}_i^j = 0$
3: **else**
4: According to equation (1), compute new weight ω_i^j based on $\hat{X}_i^j$, $\hat{Y}_i^j$, ω_i^{j-1}.
5: $n_i^j = n_i^{j-1} + \hat{n}_i^j$
6: **end if**
7: ω_i^j and n_i^j will be used in the local re-aggregation.

Algorithm 3 Local Reaggregation for Client i

Input: gw_m, **lab** (the label set), L (the number of received clients), N_A^m, ω_i^j (the new weight of client i), and n_i^j (the sample number of clients).
Output: ${}^c\omega_i^j$ and $\bar{N}_i^j$, U_i^j.

1: **Initialization:** The last upload time is t_i^l, $\omega_i^{t_i^l}$ is the uploaded weight of client i at time t_i^l, and the client number in local re-aggregation $U_i^j = 0$.
2: **if** the client don't receive global weight from server **then**
3: ${}^c\omega_i^j = \omega_i^j$ and $\bar{N}_i^j = n_i^j$ (A_0)
4: **else if lab** includes lab_i **then**
5: ${}^c\omega_i^j = (gw_m \times N_A^m - n_i^{t1} \times \omega_i^{t1} + n_i^j \times \omega_i^j)/(N_A^m - n_i^{t_i^l} + n_i^j)$, $\bar{N}_i^j = (N_A^m - n_i^{t_i^l} + n_i^j)$ and $U_i^j = L$ (A_1)
6: **else if lab** does not includes lab_i **then**
7: ${}^c\omega_i^j = (gw_m \times N_A^m + n_i^j \times \omega_i^j)/(N_A^m + n_i^j)$, $\bar{N}_i^j = (N_A^m + n_i^j)$ and $U_i^j = L + 1$ (A_2)
8: **end if**

To address this issue, we propose a local reaggregation strategy, as shown in Algorithm 3. The key idea of the strategy is to reaggregate the global weight gw_m and the new local weight ω_i^j in the client. In order to obtain a high prediction accuracy of ${}^c\omega_i^j$, we need to consider two cases.

1) If the client participates in the aggregation of gw_m, it will lead to bias, because gw_m and ω_i^j are both associated with the same datasets. This bias may result in unfairness in the learned model [28]. Here, we develop a parameter elimination method. Specifically, the relatively stale weight of the client in the global weight is deleted, so that the bias is eliminated. To achieve this purpose, first, we need to track the client's weight to determine whether there will be bias. Here, the label of each client is defined as $lab_i = [id, t_i^l]$, where id is a randomly selected temporary identifier of the client and t_i^l represents the last upload time of client i. These labels will travel between the clients and the cloud server with uploaded updates and global weights. Then, the latest weight $\omega_i^{t_i^l}$ and the number of samples $n_i^{t_i^l}$ that have been received by the cloud are saved in clients. When the client i receives lab that includes lab_i, we delete the client's local weight form the global weight by subtracting the $n_i^{t_i^l} \times \omega_i^{t_i^l}$, as shown in A_1 of Algorithm 3.
2) If the client does not participate in the aggregation of gw_m, the client's local weight ω_i^j can be reaggregated directly, as shown in A_2 of Algorithm 3. Finally, the calculated reaggregated values ${}^c\omega_i^j$, $\bar{N}_i^j$, (i.e., the total sample number in ${}^c\omega_i^j$) and U_i^j (i.e., the total client number in ${}^c\omega_i^j$) will be used in the weighted processing except to the results of A_0. In addition, lab_i will be renewed after A_1 or A_2.

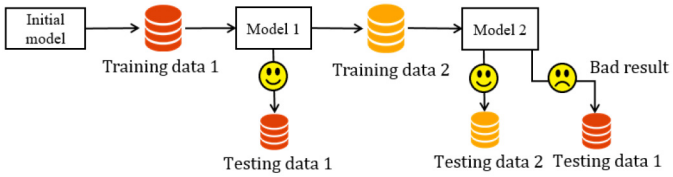


Fig. 6. Catastrophic forgetting problem illustration.

C. Weighted Processing Strategy

The weighted processing strategy is developed to overcome the catastrophic forgetting problem caused by dynamic clients. When trained on one dataset, then trained on a second dataset, many machine-learning models will "forget" the knowledge learned from the first dataset, so a catastrophic forgetting problem occurs. An example is shown in Fig. 6 to illustrate the catastrophic forgetting problem. Model 1 is generated based on training over training data 1. Based on Model 1, continuous learning is performed by using the new dataset 2, and finally, Model 2 is generated. Due to the catastrophic forgetting problem, Model 2 has a bad prediction accuracy on testing data 1, which means that Model 2 forgets the knowledge from the previous learning.

In FCL-BL, the clients are allowed to join and leave at any time, so the number of clients is not fixed in each round. Moreover, the client's datasets are changed over time, so the new training datasets (includes *bs* client's training dataset) can be significantly different from the ones in the previous rounds. Thus, FCL-BL suffers from a catastrophic forgetting problem. At first glance, the catastrophic forgetting problem can be prevented if the historic datasets and new datasets are jointly used in continuous learning. Unfortunately, however, it is difficult to collect all historic datasets and newly produced datasets of the *bs* clients who participate in the training, because each client only knows its own datasets in the setting of FL.

In this article, we propose a weighted strategy to address the catastrophic forgetting problem. On receipt of the global weight from the cloud server, the weighted processing strategy allows each client to compute the weighted average of the global weight and the client's local weight. Assigning an appropriate weight will help to remember the previous knowledge while avoiding underfitting or overfitting the new datasets. In this article, $({}^c\omega_i^j \times \bar{N}_i^j - n_i^j \times \omega_i^j)/(\bar{N}_i^j - n_i^j)$ is used to represent the global weight without bias, and $(\bar{N}_i^j - n_i^j)/[(U_i^j - 1) \times n_i^j]$ is taken as the assigned weight. Thus, the weighted average $\bar{\omega}_i^j$ (i.e., client's update) can be written as $\bar{\omega}_i^j = (({}^c\omega_i^j \times \bar{N}_i^j - n_i^j \times \omega_i^j)/[n_i^j \times (U_i^j - 1)] + \omega_i^j$. Finally, $(\bar{\omega}_i^j, n_i^j)$ is uploaded to the server for next aggregation.

VII. Analysis

In this section, we analyze the parameter setting, the impact of local reaggregation, and the weighted processing strategy.

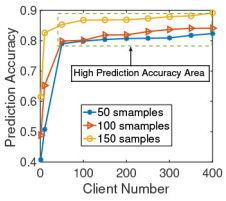


Fig. 7. *bs*: batch size.

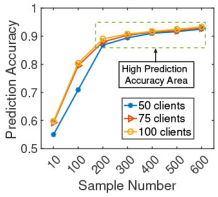


Fig. 8. thr_1: sample threshold.

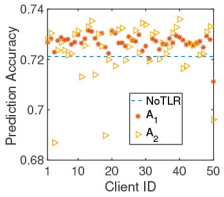


Fig. 9. Local reaggregation.

A. Parameter Analysis and Setting

By properly adjusting the parameters *bs* and thr_1, the efficiency of communication and local training in FCL-BL will be improved greatly. An appropriate batch size *bs* and sample number thr_1 should be small while without sacrificing the prediction accuracy of the global model. The parameter analysis is based on some experiments for the FL scheme, which are conducted on the MNIST dataset (the experimental setting is the same as described in Section VIII).

Fig. 7 exhibits the prediction accuracy as the function of the client number. The experimental results show that the aggregation with 50 client's updates or more provides better prediction accuracy. Hence, we can set the batch size *bs* to 50. It can reduce the aggregation time and the number of client–server interaction rounds while maintaining the high prediction accuracy of the global model. For the sample threshold thr_1, Fig. 8 shows the prediction accuracy as the function of the sample number. The experimental results indicate that the prediction accuracy is high with 200 or more samples per client. Therefore, the sample threshold thr_1 is preferably 200 to ensure the high prediction accuracy of local training.

B. Impact of Local Reaggregation Strategy

The local reaggregation improves the prediction accuracy of the global model. To verify it, we compare the weights ${}^c w_i^j$ and the global weight W_{LR} with gw_m without local reaggregation, where W_{LR} is the global weight based on ${}^c w_i^j$. Then, we have the following claim.

Claim 1: The model updated by ${}^c w_i^j$ and W_{LR} provides higher prediction accuracy than that by gw_m.

Validation: More training data always means a higher prediction accuracy. The local reaggregation just follows this idea. To verify the impact of local reaggregation, we conduct the experiments on A_1, A_2, and NoTLR (gw_m is model weights) in 50 clients, where each client has 100 samples. A_1 and A_2 represent two different cases in the local reaggregation strategy. In Fig. 9, the clients' prediction accuracy of the local model with A_1 and A_2 is higher than that of without local reaggregation in most instances. Furthermore, to compare the prediction accuracy of the global model after local reaggregated. The experiments on the different new samples and different percentages of clients who participate in local reaggregation are conducted. The prediction accuracy of the global model after A_1, A_2, Hybrid method (includes A_1 and A_2), and NoTLR are presented in Fig. 10, where the global weights of A_1, A_2 and the Hybrid method represent the three styles of W_{LR}. From the experimental results, the prediction accuracy of local model after local reaggregation is higher. Therefore, Claim 1 is validated.

In summary, the local reaggregation strategy improves training accuracy, and then the local model updated by ${}^c w_i^j$ will provide higher prediction accuracy.

C. Impact of Weighted Processing Strategy

The weighted processing strategy is feasible under some extreme conditions and superior to the method by using a weighted processing strategy in the server. h_1, h_2, and h_3 are represented as the set of clients using A_1, A_0, and A_2 of Algorithm 3, respectively. The set of clients in the round $m+1$ of FCL-BL is defined as $H = h_1 \cup h_3$. The relationship between the set of clients $L = h_1 \cup h_2$ in round m and H in round $m+1$ is described in Fig. 11.

gw_{m+1} represents the global weight of round $m+1$ in FCL-BL and W_C is global weight based on ω_i^j. Then, we have

$$
\begin{aligned}
W_C &= \sum_{i\in H} \omega_i^j n_i^j \Big/ \sum_{i\in H} n_i^j \\
gw_{m+1} &= \sum_{i\in H} \bar{\omega}_i^j n_i^j \Big/ \sum_{i\in H} n_i^j \\
&= \sum_{i\in H} \left(\frac{{}^c\omega_i^j \cdot \bar{N}_i^j - n_i^j \cdot \omega_i^j}{U_i^{t_i^l} - 1} \omega_i^j \cdot n_i^j \right) \Big/ \sum_{i\in H} n_i^j \\
&= \left[\frac{\sum_{i\in h_1} \left(gw_m \cdot N_A^m - \omega_i^{t_i^l} n_i^{t_i^l} \right)}{|L| - 1} + \frac{\sum_{i\in h_3} \left(gw_m \cdot N_A^m \right)}{|L|} \right] \\
&\quad \Big/ \sum_{i\in H} n_i^j + W_C \\
&= \left[\left(\frac{|h_1|}{|L| - 1} + \frac{|h_3|}{|L|} \right) \cdot gw_m \cdot N_A^m - \frac{1}{|L| - 1} \sum_{i\in h_1} \omega_i^{t_i^l} n_i^{t_i^l} \right] \\
&\quad \Big/ \sum_{i\in H} n_i^j + W_C \\
&= \left[\left(\frac{|h_1| - 1|}{|L| - 1} + \frac{|h_3|}{|L|} \right) \sum_{i\in h_1} \omega_i^{t_i^l} n_i^{t_i^l} + \left(\frac{|h_1|}{|L| - 1} + \frac{|h_3|}{|L|} \right) \right.
\end{aligned}
$$

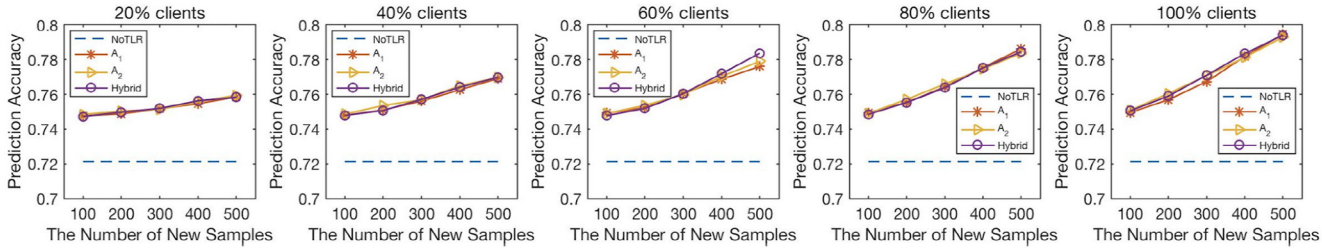


Fig. 10. Prediction accuracy of the local model after A_1, A_2, hybrid method, and NoTLR.

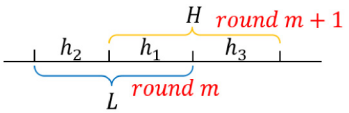


Fig. 11. Set of clients for aggregation in two adjacent rounds.

$$\times \sum_{i\in h_2} \omega_i^{t_i^l} n_i^{t_i^l} \Bigg] \Big/ \sum_{i\in H} n_i^j + W_C.$$

When $|h_1| = |L|, |h_3| = |h_2| = 0$, or $|h_3| = |L|, |h_1| = |h_2| = 0$, $gw_{m+1} = \sum_{i\in L} \omega_i^{t_i^l} n_i^{t_i^l} / \sum_{i\in H} n_i^j + W_C$. The new client's updates are used for the next round of local training, which is similar to an iterative operation if the data are static.

The method by using a weighted processing strategy in the server to tackle the catastrophic forgetting problem is presented as follows.

M1 (Weighted Processing Strategy in Cloud Server): Clients train new samples to obtain new updates ω_i^j, which will be uploaded to the cloud server. Then, in the server, we compute the weighted average of the new updates and gw_m by using $a \times \sum_{i=1}^{H} \omega_i^j + b \times gw_m$, where $a = \sum n_i^j/|H|$ and $b = \sum n_i^{t_i^l}/|L|$ represent the average of the received samples on them. The weights a and b are used to keep the balance between the previous knowledge and the knowledge on new datasets.

For M1, the global weight of the round $m+1$ is $W_{M1} = (|H|/|L|) \sum_{i=1}^{L} \omega_i^{t_i^l} \cdot n_i^{t_i^l} / \sum_{i=1}^{H} n_i^j + W_C$. In the proposed weighted processing strategy, the global weight of the round $m+1$ is $gw_{m+1} = [([(h_1 - 1)/(L-1)] + (h_3/L)) \sum_{i\in h_1} \omega_i^{t_i^l} n_i^{t_i^l} + ([|h_1|/(|L|-1)] + [|h_3|/|L|]) \times \sum_{i\in h_2} \omega_i^{t_i^l} n_i^{t_i^l}] / \sum_{i\in H} n_i^j + W_C$. Define the global weight based on the updates from all previously received clients and the clients in the next round as W_{global}, which can reflect the training results most directly and accurately. W_{global}, W_{M1}, and gw_{m+1} are described as follows:

$$W_{\text{global}} = \left(\sum_{i\in h_1\cup h_3} \omega_i^j n_i^j + \sum_{i\in h_2} \omega_i^{t_i^l} \cdot n_i^{t_i^l} \right) \Big/ \left(\sum_{i\in H} n_i^j + \sum_{i\in h_2} n_i^{t_i^l} \right)$$

$$W_{M1} = \left(\frac{|H|}{|L|} \sum_{i\in L} \omega_i^{t_i^l} n_i^{t_i^l} + \sum_{i\in h_1} \omega_i^j n_i^j + \sum_{i\in h_3} \omega_i^j n_i^j \right) \Big/ \sum_{i\in H} n_i^j$$

$$gw_{m+1} = \Bigg[\left(\frac{|h_1| - 1|}{|L| - 1} + \frac{|h_3|}{|L|} \right) \sum_{i\in h_1} \omega_i^{t_i^l} n_i^{t_i^l} + \left(\frac{|h_1|}{|L| - 1} + \frac{|h_3|}{|L|} \right) \times \sum_{i\in h_2} \omega_i^{t_i^l} n_i^{t_i^l} + \sum_{i\in h_1} \omega_i^j n_i^j + \sum_{i\in h_3} \omega_i^j n_i^j \Bigg] \Big/ \sum_{i\in H} n_i^j.$$

Claim 2: gw_{m+1} is more similar to W_{global} than W_{M1}, and provides higher prediction accuracy.

Validation: For W_{global}, the distribution in $\{W_C, \sum_{i\in h_1} \omega_i^{t_i^l} n_i^{t_i^l}, \sum_{i\in h_2} \omega_i^{t_i^l} n_i^{t_i^l}\}$ equals $[\sum_{i\in H} n_i^j : 0 : 1]$. Similarly, the distributions of W_{M1} and gw_{m+1} are $[\sum_{i\in H} n_i^j : (|H|/|L|) : (|H|/|L|)]$ and $[\sum_{i\in H} n_i^j : ([|h_1| - 1|/(|L| - 1)] + (|h_3|/|L|)) : ([|h_1|/(|L| - 1)] + (|h_3|/|L|))]$, respectively.

1) When $|h_1| = |L|$, we obtain $|h_2| = 0$. The distribution of W_{M1} and gw_{m+1} is both $[\sum_{i\in H} n_i^j : 1 : 0]$.
2) When $|h_1| = 0$, we obtain $|h_3| = |L|$. The distribution of W_{M1} and gw_{m+1} is both $[\sum_{i\in H} n_i^j : 0 : (|h_3|/|L|)]$.

To test the prediction accuracy of the global model, experiments on different new samples are conducted. The right-most two subfigures of Fig. 12 show the above two cases. The prediction accuracy of the global model updated by W_{M1} and gw_{m+1} are the same because they have the same distribution. Specially, when $|h_3| = |L|$, the distribution of W_{global} is the same as W_{M1} and gw_{m+1}, the right-most subfigure shows the result.

In the other condition, that is, $|h_1| > 0$ and $|h_3| > 0$, we use the Euclidean distance to calculate the similarity of the distributions of W_{global}, W_{M1}, and gw_{m+1}. For the sake of simplicity, the distributions of W_{global}, W_{M1}, and gw_{m+1} are represented as R_g, R_{M1}, and R_A, respectively. Then, their Euclidean distance can be written as $d(R_g, R_{M1}) = \sqrt{(|H|/|L|)^2 + ([|H|/|L|] - 1)^2}$ and

$$d(R_g, R_A) = \sqrt{\left(\frac{|h_1| - 1}{|L| - 1} + \frac{|h_3|}{|L|} \right)^2 + \left(\frac{|h_1|}{|L| - 1} + \frac{|h_3|}{|L|} - 1 \right)^2}$$

where $|H| = |L|$ in the setting of FCL-BL. Then, according to $d(R_g, R_{M1})$ and $d(R_g, R_A)$, we have

$$\begin{aligned} &d(R_g, R_{M1})^2 - d(R_g, R_A)^2 \\ &= 1 - \left[\left(\frac{|h_1| - 1}{|L| - 1} + \frac{|h_3|}{|L|} \right)^2 + \left(\frac{|h_1|}{|L| - 1} + \frac{|h_3|}{|L|} - 1 \right)^2 \right] \\ &= 1 - \frac{(|H|^4 - 2|H|^3 - (2h_3 - 2)|H|^2 + 2|h_3|^2}{(|H| - 1)^2 |H|^2} \end{aligned}$$

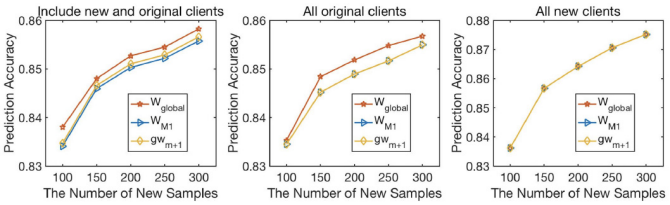


Fig. 12. Accuracy comparison in W_{global}, W_{M1}, and gw_{m+1}.

$$= \frac{(2|h_3|-1)|H|^2 - 2|h_3|^2}{(|H|-1)^2|H|^2}$$

where $|h_3| > 0$, $|H| = |L| > h_3$, and they are all integers. Then, it always holds $[d(R_g, R_{M1})^2 - d(R_g, R_A)^2] > 0$. As $d(R_g, R_{M1}) > 0$ and $d(R_g, R_A) > 0$, we obtain $d(R_g, R_{M1}) > d(R_g, R_A)$. That means gw_{m+1} is more similar to W_{global} than W_{M1} in the distribution. The more similar, the higher prediction accuracy, and the left-most subfigure of Fig. 12 also verifies the conclusion. Hence, Claim 2 is validated.

VIII. Performance Evaluation

In this section, we first introduce experiment datasets, performance metrics, and implementation. Then, we evaluate the performance of FCL-BL and compare it with the other three schemes (Fed-APD [23], Fed-Restart, and Fed-Regularizer).

A. Datasets, Metrics, and Implementation

1) *Datasets:* a) The MNIST handwritten digits dataset[1] consists of 60 000 training images (28 × 28 pixels) and 10 000 testing images with ten categories (i.e., labels). b) The NORB dataset[2] contains 24 300 training samples and 24,300 testing samples with five generic categories. Each sample includes an image pair with $2 \times 32 \times 32$ pixels. c) The FASHION dataset[3] has 44 000 color images with six categories and $3 \times 60 \times 80$ pixels. 80% of the dataset is taken as training samples and the other 20% is testing samples. d) The SIGNAL dataset is collected from four universal software radio peripheral (USRP) platforms, including 4000 samples. Each sample has 1024×16 data points. The examples of MNIST, NORB, and FASHION datasets and the collection of the SIGNAL dataset are shown in Fig. 13.
Metrics: Three metrics are used: a) overhead; b) prediction accuracy; and c) efficiency. The overhead is involved in communication volume and data upload. The prediction accuracy is evaluated in three aspects: a) quantity imbalance and Non-IID setting; b) a round of FL; and c) continuous learning. The efficiency of FCL-BL is reflected by training time, and lower training time means higher efficiency.

2) *Implementation:* FCL-BL implementations are achieved by PYTHON 3.7 on PC with 3.1-GHz Intel Core i5, 8-GB main memory, and macOS High Sierra operating system. The FL schemes compared are based on the convolutional neural network (CNN) with two 5×5 convolution layers, two fully connected layers, and ReLu activation, which are the typical settings used in [8]. The BL model used in FCL-BL is based on 100 feature nodes and 3000 enhancement nodes and also uses ReLu activation.

[1]http://yann.lecun.com/exdb/mnist/
[2]https://cs.nyu.edu/~ylclabylclab/data/norb-v1.0-small/
[3]https://www.kaggle.com/paramaggarwal/fashion-product-images-small

Fig. 13. Examples for training figures. (a) MNIST. (b) NORB. (c) FASHION. (d) SIGNAL.

TABLE III
Size of Uplink

Model \ Datasets	MNIST	NORB	FASHION
FedAvg-based Schemes	21750	30750	799194
FCL-BL	31000	15500	18600

B. Experimental Results

We evaluate the performance of FCL-BL in terms of overhead, prediction accuracy, and efficiency.

1) *Overhead:* The communication between clients and server is expensive and the data uplink is time consuming. The regular FL schemes, such as [7]–[12], always require multiple rounds of aggregation to achieve convergence. For faster convergence and less client–server communication volume, the direct approach is to learn as much as possible from local data samples in one round of FL. Unfortunately, this approach can result in more weight divergence between different clients, and then decreases the aggregation effect of the global model. However, FCL-BL can eliminate this weight divergence, which means fewer communication rounds are achieved. For the practical experiments on the MNIST, NORB, and FASHION datasets, the size of the model weights uploaded to the server in each client is shown in Table III. Fewer data are transmitted in FCL-BL when the samples are images with large pixels. In summary, the fewer communication rounds and data uplink for the images of large pixels reduce the overhead of FCL-BL.

2) *Prediction Accuracy:* The evaluation of prediction accuracy in the imbalanced datasets, a round of the FL, and the continuous learning shows the effectiveness of FCL-BL to handle quantity imbalance and Non-IID datasets, inaccurate-training, and incremental learning.

1) FCL-BL for Quantity Imbalance and Non-IID Datasets: The datasets in the comparison experiments satisfy two-class Non-IID, and the quantity imbalance is generated with the

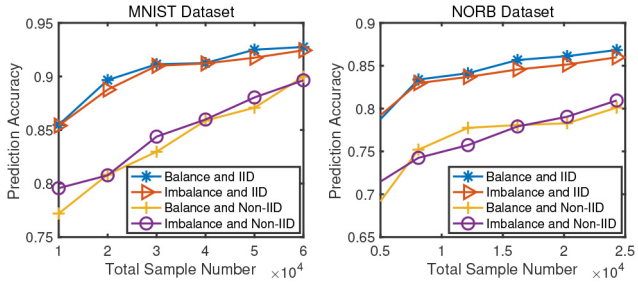


Fig. 14. FCL-BL for quantity imbalance and Non-IID data.

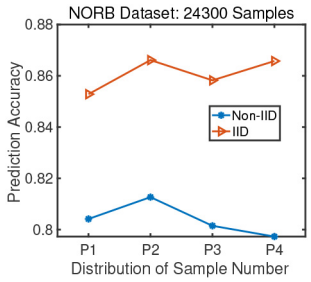


Fig. 15. FCL-BL in the imbalanced data with different extent.

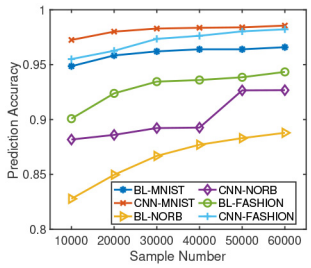


Fig. 16. Training in a centralized learning scenario.

same probability distribution. Fig. 14 shows the prediction accuracy of FCL-BL on 100 clients with a different total sample number. The results on MNIST and NORB datasets indicate that the FCL-BL is suitable for tackling the Non-IID dataset regardless of whether the quantity of client's samples is imbalanced or balanced.

Fig. 15 exhibits prediction accuracy as a function of the extent of quantity imbalance. The extent is reflected by four different probability distributions, as shown below. (Assume that there are n clients and $\{1, \ldots, n\}$ represents the n clients.).

1) *P1:* The probability of each sample being randomly assigned to each client is the same, that is, $1/n$.
2) *P2:* The probability distribution of each sample being randomly assigned to the client in $\{1, n/2\}$ and $\{1 + n/2, n\}$ is $\{2/3n, 4/3n\}$, respectively.
3) *P3:* The probability distribution of each sample being randomly assigned to the client in $\{1, n/4\}$, $\{1 + n/4, n/2\}$, $\{1 + n/2, 3n/4\}$, and $\{1 + 3n/4, n\}$ is $\{2/5n, 4/5n, 6/5n, 8/5n\}$, respectively.
4) *P4:* The probability distribution of each sample being randomly assigned to the client in $\{1, n/5\}$, $\{1 + n/5, 2n/5\}$, $\{1+2n/5, 3n/5\}$, $\{1+3n/5, 4n/5\}$, and $\{1+4n/5, n\}$ is $\{1/3n, 2/3n, 1/n, 4/3n, 5/3n\}$, respectively.

The experimental results in Fig. 15 validate FCL-BL can work well in the quantity imbalance to a different extent because the prediction accuracy of FCL-BL is very approximate in the four different probability distributions.

2) FCL-BL Handles Inaccurate-Training: The prediction accuracy of CNN is higher than that of BL in the centralized learning scenario, as shown in Fig. 16. However, in a round of FL, the FedAvg-based schemes are inaccurate, especially in the Non-IID setting, as shown in Figs. 17 and 18. The FedAvg-10, FedAvg-50, and FedAvg-75 represent the SGD with ten local epochs, 50 local epochs, and 75 local epochs, respectively. In the IID setting, the prediction accuracy of FCL-BL is similar to that of FedAvg-50 and FedAvg-75 schemes in MNIST and FASHION datasets, but 40% higher in the NORB dataset, shown in Fig. 17. In the Non-IID setting, the experimental results in Fig. 18 show the prediction accuracy of FCL-BL is 20%–60% higher than that of FedAvg-based schemes. Therefore, the proposed FCL-BL handles the inaccurate-training issue well.

3) FCL-BL for Continuous Learning: Continuous learning means more than one round of FL is performed. The incremental learning capability is reflected by overcoming the catastrophic forgetting problem and high testing accuracy.

Catastrophic Forgetting: The experiments on catastrophic forgetting are conducted in two adjacent rounds on the MNIST dataset. In each round, there are 100 clients and each client owns 300 samples (training data 1) for current training and 300 new samples (training data 2) in the next round. The new samples are with different ratios of new classes: 0%, 25%, 50%, 75%, and 100%. For example, if the ratio is 50%, the classes of 150 samples in the next round are new classes that do not appear in the previous round. In Fig. 19, the left-most bar is the testing accuracy in the current round, and others represent the results in the next round. The results in Fig. 19(a) show that the testing accuracy of the FedAvg-based scheme gradually decreases as the ratio of new classes in the next round increases. It indicates the catastrophic forgetting problem exists in regular FL. However, FCL-BL maintains high testing accuracy no matter what the ratios of classes are, as shown in Fig. 19(b). Thus, FCL-BL can overcome the catastrophic forgetting problem in continuous learning.

Comparison With Other Schemes: We compare FCL-BL with some other schemes (Fed-APD, Fed-Restart, and Fed-Regularizer) that can be used to solve the catastrophic forgetting problem. Fed-Restart retrains all client's data from the beginning in different rounds, which is the most direct method to overcome the catastrophic forgetting problem in continuous learning. Fed-APD and Fed-Regularizer are introduced in Section III.

These comparison experiments are conducted in five continuous rounds (or timestamps) on the FASHION dataset and SIGNAL dataset. The experiment settings are shown as follows.

1) 100 clients participate in training in each timestamp.
2) In the FASHION dataset, 35 200 samples are sorted by class and evenly divided into 500 partitions. Then, randomly assign five partitions to each client as the samples in five different timestamps.

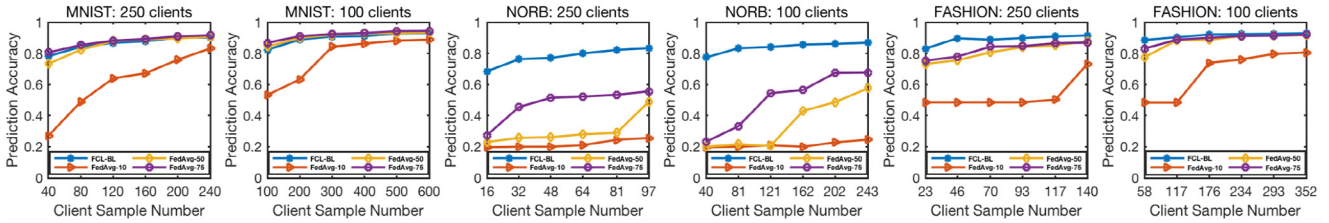


Fig. 17. Accuracy comparison for FCL-BL and FedAvg-based schemes with IID.

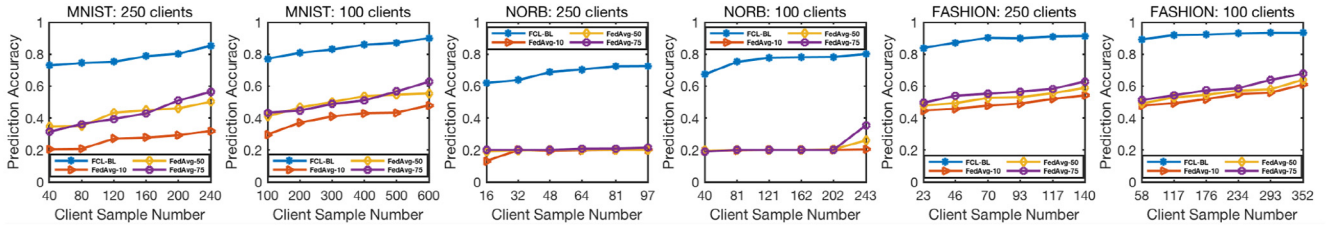


Fig. 18. Accuracy comparison for FCL-BL and FedAvg-based schemes with Non-IID.

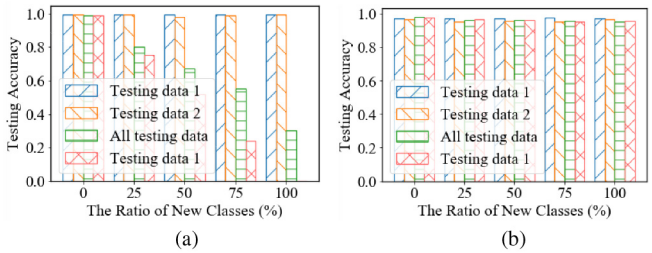


Fig. 19. (a) Catastrophic forgetting in the continuous learning. (b) FCL-BL without catastrophic forgetting.

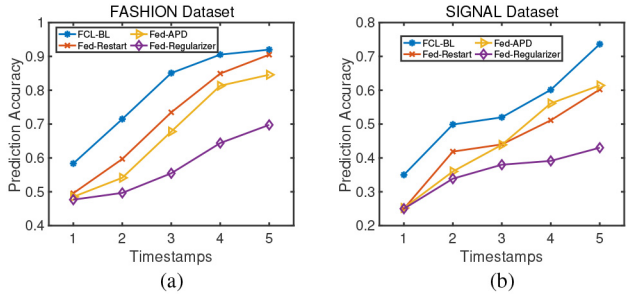


Fig. 20. Prediction accuracy comparison of continuous learning on (a) FASHION dataset and (b) SIGNAL dataset.

3) The SIGNAL dataset is natural streaming collected from the real world. We adjust the data collection time to obtain the samples in different timestamps. In each timestamp, approximately 160 samples per client are trained.

In Fig. 20, the prediction accuracy is increased over time. It indicates the four schemes can perform incremental learning. Fig. 20(a) shows that the prediction accuracy of FCL-BL is about 10% higher than that of Fed-APD, about 20% higher than that of the Fed-Regularizer, and close to that of Fed-Restart in the fifth timestamp. In Fig. 20(b), the prediction accuracy of Fed-Restart is similar to that of Fed-APD, but the prediction accuracy of Fed-Restart and Fed-APD is about 12% lower than that of FCL-BL. Because the Fed-APD and the Fed-Restart cannot handle catastrophic forgetting caused by dynamic client changes, which reduces prediction accuracy. In summary, the prediction accuracy of FCL-BL is higher than that of other schemes, indicating better incremental learning of FCL-BL.

Efficiency: The results in Fig. 21(a) show the training time on the FASHION dataset in continuous learning. The training time of FCL-BL is about 11.0 s, which is near 1.5 times faster than that of the Fed-APD and Fed-Regularizer. In addition, the training time of Fed-Restart is increased over time and far higher than that of FCL-BL.

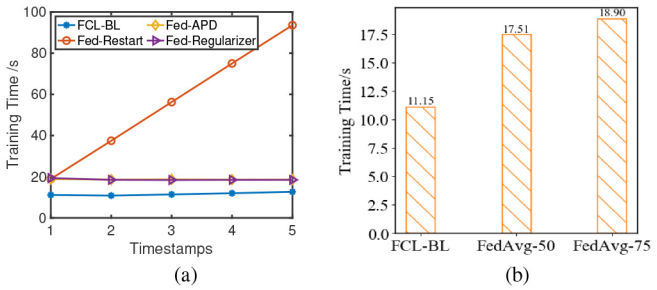


Fig. 21. (a) Training time comparison in the continuous learning. (b) Training time comparison with the same prediction accuracy.

To further verify the efficiency of FCL-BL, we compare the training time of FCL-BL with FedAvg-based schemes (FedAvg-50 and FedAvg-75) when they achieve the same prediction accuracy (92.0% in the experiment). The training time is shown in Fig. 21(b), and the results indicate that the training time of FCL-BL is lower than that of FedAvg-50 and FedAvg-75 in a round of FL. Thus, if the Fed-Restart and the Fed-Regularizer are based on FedAvg-50, the FCL-BL is also more efficient in continuous learning.

IX. Conclusion

In this article, we have designed an efficient and accurate FCL-BL. In FCL-BL, we develop a local-independent training solution and a batch-asynchronous approach to support the fast and accurate global weight update. Based on it, a weighted processing strategy is designed to solve the catastrophic forgetting problem, so that FCL-BL can achieve continuous learning. Then, we introduce BL to tackle the computation-efficient issue in local training while avoiding the catastrophic forgetting problem in clients. In addition, to ensure the local model can provide good services to clients, we design a local reaggregation strategy. Finally, the experimental results also demonstrated that FCL-BL keeps the high prediction accuracy of the global model in the unbalanced data and Non-IID data.

In FCL-BL, there are interesting problems that deserve further investigation, such as the privacy issues in the transmission of updates and the improvement of accuracy and efficiency in FCL. For the privacy issues, the methods based on differential privacy [35], [36]; secure multiparty computation [37], [38]; homomorphic encryption [39]; and over-the-air computation [40] are used to additionally improve privacy protection. In terms of performance improvement, the fuzzy BLS in [41] is a solution to improve the prediction accuracy and training time in centralized and distributed learning, and the incremental method in [42] is used to overcome the catastrophic forgetting problem. Combined with these methods, we may further improve the performance of FCL. The aforementioned problems are left for our future work.

References

[1] J. Du, C.-M. Vong, and C. L. P. Chen, "Novel efficient RNN and LSTM-like architectures: Recurrent and gated broad learning systems and their applications for text classification," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1586–1597, Mar. 2021.

[2] D. Liu and S. Yue, "Event-driven continuous STDP learning with deep structure for visual pattern recognition," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1377–1390, Apr. 2019.

[3] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24. Burlington, VT, USA: Elsevier, 1989, pp. 109–165.

[4] S. Zheng *et al.*, "Asynchronous stochastic gradient descent with delay compensation," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 4120–4129.

[5] T. D. Bui, C. V. Nguyen, S. Swaroop, and R. E. Turner, "Partitioned variational inference: A unified framework encompassing federated and continual learning," 2018. [Online]. Available: arXiv:1811.11206.

[6] C. L. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[7] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," 2015. [Online]. Available: arXiv:1511.03575.

[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[9] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018. [Online]. Available: arXiv:1806.00582.

[10] X. Li, W. Yang, S. Wang, and Z. Zhang, "Communication efficient decentralized training with multiple local updates," 2019. [Online]. Available: arXiv:1910.09126.

[11] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019. [Online]. Available: arXiv:1902.01046.

[12] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," 2019. [Online]. Available: arXiv:1905.10497.

[13] J. Lou and Y.-M. Cheung, "Uplink communication efficient differentially private sparse optimization with feature-wise distributed data," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 125–133.

[14] J. Lou and Y.-M. Cheung, "An uplink communication-efficient approach to featurewise distributed sparse optimization with differential privacy," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 17, 2020, doi: 10.1109/TNNLS.2020.3020955.

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

[16] L. Jiao, N. Wang, P. Wang, A. Alipour-Fanid, J. Tang, and K. Zeng, "Physical layer key generation in 5G wireless networks," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 48–54, Oct. 2019.

[17] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[18] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019. [Online]. Available: arXiv:1903.03934.

[19] M. R. Sprague *et al.*, "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2018, pp. 21–28.

[20] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," 2019. [Online]. Available: arXiv:1911.02134.

[21] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[22] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017.

[23] J. Yoon, W. Jeong, G. Lee, E. Yang, and S. J. Hwang, "Federated continual learning with adaptive parameter communication," 2020. [Online]. Available: arXiv:2003.03196.

[24] K. Yang, T. Fan, T. Chen, Y. Shi, and Q. Yang, "A Quasi-Newton method based vertical federated learning framework for logistic regression," 2019. [Online]. Available: arXiv:1912.00513.

[25] C. He *et al.*, "FedML: A research library and benchmark for federated machine learning," 2020. [Online]. Available: arXiv:2007.13518.

[26] F. Xu, Z. Pan, and R. Xia, "E-commerce product review sentiment classification based on a naïve bayes continuous learning framework," *Inf. Process. Manag.*, vol. 57, no. 5, 2020, Art. no. 102221.

[27] J. Huang *et al.*, "Cross-language transfer learning, continuous learning, and domain adaptation for end-to-end automatic speech recognition," 2020. [Online]. Available: arXiv:2005.04290.

[28] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019. [Online]. Available: arXiv:1912.04977.

[29] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1310–1321.

[30] S. Thrun, "A lifelong learning perspective for mobile robot control," in *Intelligent Robots and Systems*. Amsterdam, The Netherlands: Elsevier, 1995, pp. 201–214.

[31] M. Han, S. Feng, C. L. P. Chen, M. Xu, and T. Qiu, "Structured manifold broad learning system: A manifold perspective for large-scale chaotic time series analysis and prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 9, pp. 1809–1821, Sep. 2019.

[32] M. Xu, M. Han, C. L. P. Chen, and T. Qiu, "Recurrent broad learning systems for time series prediction," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1405–1417, Apr. 2020.

[33] S. Feng and C. P. Chen, "Broad learning system for control of nonlinear dynamic systems," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2018, pp. 2230–2235.

[34] J. Jin, Z. Liu, and C. P. Chen, "Discriminative graph regularized broad learning system for image recognition," *Sci. China Inf. Sci.*, vol. 61, no. 11, 2018, Art. no. 112209.

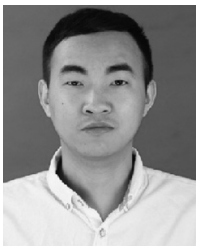
[35] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2016, pp. 308–318.

[36] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017. [Online]. Available: arXiv:1712.07557.

[37] K. Bonawitz *et al.*, "Practical secure aggregation for federated learning on user-held data," 2016. [Online]. Available: arXiv:1611.04482.

[38] K. Bonawitz, F. Salehi, J. Konečnỳ, B. McMahan, and M. Gruteser, "Federated learning with autotuned communication-efficient secure aggregation," in *Proc. 53rd Asilomar Conf. Signals Syst. Comput.*, 2019, pp. 1222–1226.

[39] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017. [Online]. Available: arXiv:1711.10677.
[40] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.
[41] S. Feng and C. L. P. Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 414–424, Feb. 2020.
[42] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4652–4662.



Junqing Le received the B.S. degree in software engineering from Southwest Jiaotong University, Chengdu, China, in 2014, and the M.S. degree in signal and information processing from Southwest University, Chongqing, China, in 2017, where he is currently pursuing the Ph.D. degree in intelligent computing and information processing.

From 2019 to 2021, he was a visiting scholar with George Mason University, Fairfax, VA, USA. His research interests include privacy protection, privacy machine learning, and cloud computing security.

Xinyu Lei (Member, IEEE) received the B.S. and M.S. degrees from the Department of Computing Science, Chongqing University, Chongqing, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA.

He worked with Texas A&M University at Qatar, Doha, Qatar, and Ford Motor Company, Dearborn, MI, USA, as a Research Assistant in 2013 and 2017, respectively. His current research focuses on networks, IoT, cloud computing, database, and blockchain.

Nankun Mu (Member, IEEE) received the B.S. degree in software engineering, the M.S. degree in computer systems and structures, and the Ph.D. degree in computer science and technology from Chongqing University, Chongqing, China, in 2011, 2013, and 2015, respectively.

His research areas include multiagent control systems, evolutionary computation, and information security.

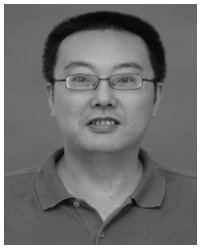
Hengrun Zhang received the B.S. degree in automation from the East China University of Science and Technology, Shanghai, China, in 2012, the M.S. degrees in control science and engineering from Shanghai Jiao Tong University, Shanghai, in 2015, and in computer science from George Mason University, Fairfax, VA, USA, in 2018, where he is currently pursuing the Ph.D. degree with the Computer Science Department.

His research interests include federated learning applications in the Internet of Things, performance optimization in federated learning, task scheduling in networking, and privacy-preserving communication.

Kai Zeng (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Worcester Polytechnic Institute, Worcester, MA, USA, (WPI), in 2008.

He was a Postdoctoral Scholar with the Department of Computer Science, University of California at Davis (UCD), Davis, CA, USA, from 2008 to 2011. He was with the Department of Computer and Information Science, University of Michigan–Dearborn, Dearborn, MI, USA, as an Assistant Professor, from 2011 to 2014. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Cyber Security Engineering, and the Department of Computer Science, George Mason University, Fairfax, VA, USA. His current research interests include cyber-physical system/IoT security and privacy, 5G and beyond wireless network security, network forensics, machine learning, and spectrum sharing.

Dr. Zeng was a recipient of the U.S. National Science Foundation Faculty Early Career Development (CAREER) Award, in 2012, the Excellence in Postdoctoral Research Award from UCD, in 2011, and the Sigma Xi Outstanding Ph.D. Dissertation Award from WPI, in 2008. He is an Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.



Xiaofeng Liao (Fellow, IEEE) received the B.S. and M.S. degrees in mathematics from Sichuan University, Chengdu, China, in 1986 and 1992, respectively, and the Ph.D. degree in circuits and systems from the University of Electronic Science and Technology of China, Chengdu, in 1997.

From 1999 to 2012, he was a Professor with Chongqing University, Chongqing, China, where he is a Professor and the Dean of the College of Computer Science. He is also a Yangtze River Scholar of the Ministry of Education of China, Chongqing University. From November 1997 to April 1998, he was a Research Associate with the Chinese University of Hong Kong, Hong Kong. From October 1999 to October 2000, he was a Research Associate with the City University of Hong Kong, Hong Kong. From March 2001 to June 2001 and March 2002 to June 2002, he was a Senior Research Associate with the City University of Hong Kong. From March 2006 to April 2007, he was a Research Fellow with the City University of Hong Kong. He holds four patents and published four books and over 300 international journal and conference papers. His current research interests include neural networks, nonlinear dynamical systems, cryptography, and privacy protection.

Dr. Liao is an Associate Editor of IEEE TRANSACTION ON CYBERNETICS and IEEE TRANSACTION ON NEURAL NETWORK AND LEARNING SYSTEMS.