SLMFed: A Stage-Based and Layerwise Mechanism for Incremental Federated Learning to Assist Dynamic and Ubiquitous IoT

Linlin You, *Senior Member, IEEE*, Zihan Guo, Bingran Zuo, Yi Chang, *Senior Member, IEEE*, and Chau Yuen, *Fellow, IEEE*

***Abstract*—Along with the vast application of Internet of Things (IoT) and the ever-growing concerns about data protection, a novel type of learning, named incremental federated learning (IFL), is rising to further elevate the intelligence and quality of various IoT systems and services by consistently learning and updating their models, e.g., deep neural networks, in dynamic contexts, where clients and data can increase and accumulate gradually. Since IFL is still in its infancy, to overcome its emerging challenges as represented in 1) periodic learning about how to initialize the model update rationally to avoid catastrophic performance dropping, and 2) iterative learning about how to update the model cost-efficiently to remedy overlearning on duplicated information, this article proposes a stage-based and layerwise mechanism for IFL, called SLMFed, in which, the periodic learning is managed by a stage transition and client selection strategy to trigger model update according to the quantitative and qualitative changes on clients, data, and user experience, and the iterative learning is enhanced by an adaptive layer uploading and aggregation strategy to update the global model by measuring representational consistencies and information richness of local model layers. As shown by the evaluation results, SLMFed can not only stabilize the learning across various learning stages but also boost the performance in terms of learning accuracy, communication cost, and stage contribution by about 32.09%, 105.94%, and 22.02%, respectively.**

***Index Terms*—Federated learning (FL), incremental federated learning (IFL), incremental learning (IL), layer uploading and aggregation, layerwise iterative learning, stage transition and client selection, stage-based periodic learning.**

Manuscript received 23 September 2023; revised 1 December 2023; accepted 6 January 2024. Date of publication 15 January 2024; date of current version 25 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62002398; in part by the GuangDong Basic and Applied Basic Research Foundation under Grant 2023A1515012895; and in part by the National Research Foundation Singapore through the AI Singapore Program (AISG) under Award AISG2-TC-2023-008-SGKR. *(Corresponding author: Yi Chang.)*

Linlin You and Zihan Guo are with the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 518107, China.

Bingran Zuo is with the Rehabilitation Research Institute of Singapore, Nanyang Technological University, Singapore 639798.

Yi Chang is with the School of Artificial Intelligence, Jilin University, Changchun 130012, China (e-mail: yichang@jlu.edu.cn).

Chau Yuen is with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore 639798.

Digital Object Identifier 10.1109/JIOT.2024.3353793

I. Introduction

With the vast deployment and utilization of interconnected smart objects, which contain rich and independent computing, storage, and communication capabilities, the fusion of ubiquitous Internet of Things (IoT) and versatile artificial intelligence (AI) is being explored to renovate modern service profiles in facilitating our daily lives [1], e.g., autonomous vehicle-enabled on-demand mobility [2], assistive robotic-driven personalized rehabilitation [3], etc. Along with this trend, diverse objects are deployed even closer to the users as digital companions that can consistently monitor the status of individuals and their surroundings, and then, timely react to meet emerging needs with high-user experience, e.g., the obstacle avoidance of unmanned vehicles while riding passengers to their destinations [4], the falling protection of assistive robotics during the movement of patients [5], etc.

To achieve such a promising vision, collective intelligence that integrates and intercepts knowledge retrieved from each object becomes essential to continuously elevate the level of automation and intelligence of pervasive IoT systems and services [6], [7], [8]. However, the ever-growing concern about data security and user privacy may overturn widely used centralized approaches in fusing interknowledge [9], [10], [11], and the gradually accumulated clients and newly sensed data may also complicate the procedure to train efficient and effective AI models, e.g., deep neural networks (DNNs), that can consolidate both old and new knowledge to avoid catastrophic performance dropping [12]. Hence, as a novel solution, incremental federated learning (IFL), also known as continuous federated learning (FL), is attracting more attention by applying FL and incremental learning (IL) jointly to learn a shareable and informed model in a privacy-preserving and change-adapting manner [13], [14], [15]. Currently, IFL solutions to tackle samples with new classes [16] are under discussion, and also utilized to support several application scenarios, e.g., person reidentification [17], computer-aided diagnosis [18], and intrusion detection in unmanned aerial vehicle networks [19].

In general, to assist real-world applications, IFL needs to run periodically, called periodic learning, and then, update the global model iteratively, called iterative learning, e.g., in rehabilitation, as the motor ability of paralytics will

recover gradually, related observations shall be processed timely for an adaptive model that can be deployed in assistive robotics to better advise and serve the patients [20]; and in mobility, as the regularity of road networks may change over time and places, the model deployed at the control center to coordinate autonomous vehicles needs to be updated consistently to resolve intertwined user demand and system supply with high-user experience and low-operation cost [21]. Even though periodic learning can adaptively optimize IoT services, and iterative learning can assist in knowledge integration of devices, it still needs to overcome challenges emerging in 1) periodic learning to start IFL rationally with a set of qualified clients (QCs) selected as learning participants, and 2) iterative learning to execute IFL cost-efficiently for an updated global model [22], [23], [24].

Since IFL is still in its infancy, current solutions focus more on static scenarios (i.e., with a fixed number of clients and data) to resolve client selection, communication optimization, and aggregation enhancement issues caused by non-independent and identically distributed (non-IID) data, heterogeneous computing capabilities, the discrepancy in local models, etc. [25], [26], [27]. While considering real-world situations, the dynamics of IFL, i.e., the gradually increased clients and local data, are rarely discussed to train AI models with balanced performance on old and new data. It is critical as time-varying contexts commonly exist that may unsettle the one-time basis of current solutions, making them less efficient or even invalidated. Therefore, a novel solution is required to support IFL under such a dynamic context cost-efficiently.

To fill the gap, this article proposes SLMFed, a stage-based and layerwise mechanism to support IFL for ubiquitous IoT systems and services. In general, the main contributions of this study can be summarized as the following.

1) *Stage-Based Periodic Learning Is Managed:* It proposes dedicated indicators measuring the changes in active users and sensed local data as well as user experience to determine the stage transition in IoT tasks rationally. Moreover, as the premise to start a new stage, a client selection probability is designed based on a self-information change indicator to activate a set of qualified IoT devices that can maintain a balance between the old and new data to avoid catastrophic forgetting and over-learning issues.
2) *Layerwise Iterative Learning Is Designed:* It optimizes the local model uploading process to be adaptive in each client, which calculates representational consistencies (RCs) of model layers after the local training to adjust their uploading frequency with communication costs (CCs) reduced in IoT networks, and also, enhances the global model aggregation function to be layerwise in the server that measures the information richness (IR) of received local parameters to update the global model with learning performance improved for IoT services.
3) *Significant Improvement Is Achieved:* Through SLMFed, IFL in dynamic and ubiquitous IoT contexts can run efficiently and effectively as shown by the evaluation results to learn DNNs for four standard data sets (i.e., Modified National Institute of Standards and Technology (MNIST), Fashion-MNIST (FMNIST), CIFAR-10, and German Traffic Sign Recognition Benchmark (GTSRB)) in two client and data incremental scenarios. Specifically, compared to other methods in all testing cases (in total 8), it can significantly improve model performance, reduce CCs and increase stage contribution by about 32.09%, 105.94%, and 22.02%, respectively.

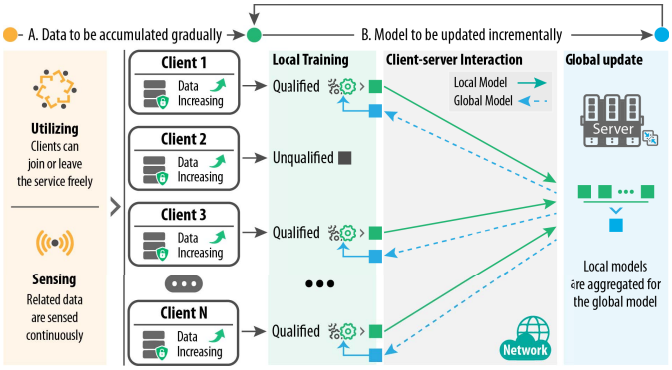


Fig. 1. Overall workflow of IFL. (a) Data are accumulated gradually while related services are in use. (b) Models are updated incrementally according to the FL paradigm, i.e., 1) Local training to learn local models at QCs, 2) Client-server interaction to exchange learning parameters and 3) Global update to aggregate received local models for the global model and then devolve the updated global model to the clients.

The remainder of this article is organized as follows. Section II introduces IFL and then, summarizes related challenges and solutions to disclose the current research gap. Next, SLMFed is presented and evaluated in Sections III and IV, respectively. Finally, Section V concludes the work and sketches the future research directions.

II. Challenges and Solutions About IFL

To establish a common research foundation, in this section, first, the overall workflow of IFL is discussed to illustrate how up-to-date AI models can be trained and deployed in a collaborative and privacy-preserving way to support various IoT systems and services. Then, several challenges encountered in IFL are identified, and related solutions are summarized to disclose the current research gap.

A. Introduction of IFL

In general, with the penetration of various IoT systems and services, massive data can be gleaned with a mixture of old and new content that reflects diversified and changeable user behaviors. Hence, to avoid the downgrade in the level of automation and intelligence for these systems and services, their employed AI models shall be updated incrementally to handle novel service contexts in time [12]. Moreover, under the ever-growing concerns of data security and user privacy, FL is rising to complement centralized approaches by training global models, e.g., DNNs, that can be shared and customized among clients in a collaborative and privacy-preserving manner [25].

As shown in Fig. 1, IFL is discussed to address the two emerging requirements jointly in two phases.

1) *Phase 1 (Data to be Accumulated Gradually):* As shown in Fig. 1(a), it represents a standby period for the number of clients and the size of data to grow. Specifically, similar to how smartphone applications are utilized, clients can choose to join or leave a service freely according to their actual needs. Hence, the cluster of clients may vary over time and places, especially for personalized services, e.g., shared mobility, on-demand healthcare, etc. Moreover, while services are in use, related objects, e.g., users, unmanned vehicles, assistive robotics, etc., are monitored to sense themself and their surroundings for essential data representing their running statuses. Accordingly, by cooperating with heterogeneous clients as well as their local data, decisive knowledge, e.g., encoded in AI models, can be learned to continuously improve service quality and user experience.
2) *Phase 2 (Model to be Updated Incrementally):* As shown in Fig. 1(b), it stands for an activation period for clients to train a global model collaboratively based on data accumulated in the first phase. According to the general workflow of FL, clients with sufficient samples are qualified and activated to start the local training, in which, each client will use its own data to train a local model. After that, local models are uploaded to the server through the network and aggregated in the server to update the global model. Finally, the global model will be distributed to the clients to start a new learning iteration or end the learning by deploying the model. It is worth noting that within the activation period, the learning is iterative, which ends when predefined conditions are met, e.g., maximum learning interactions or target model accuracy.

In summary, the incrementality of IFL can be viewed in two aspects.

1) *Periodic Learning:* It is illustrated by the switch between the standby and activation periods when the stale model needs to be updated to address previously unknown or insignificant information.
2) *Iterative Learning:* It is represented by the interaction between the clients and the server within the activation period to train a sharable and customizable global model.

To implement the two kinds of learning in IFL efficiently and effectively, several challenges are emerging.

B. Emerging Challenges

First, most of the existing research is commonly studied in an ideal environment with a fixed number of clients and size of data, and in turn, lacks the consideration of the dynamics in real-world scenarios [28], [29], [30], [31], [32]. Along with the service penetration that more active clients and representable information are attracted and sensed, respectively, two critical challenges are encountered in periodic learning.

C.1 *Initialization of Model Update:* Compared to conventional scenarios that aim to learn high-performance models in a fixed context, IFL needs to make a prudent decision on when and whether to update the staling models affected by the time-varying properties, e.g., recently recruited users, newly sensed data, etc. [33], [34].

C.2 *Selection of QCs:* User behavior may drift over time, and in turn, previously unseen or insignificant information may surge in a part of active users. Therefore, IFL needs to measure these variations, and accordingly, select appropriate clients to learn a collective model that can serve all the clients without a catastrophic dropping in user experience [9], [35].

Once IFL is initialized, related clients shall be orchestrated cost-efficiently to update the model for balanced performance in handling both old and new content. In general, within iterative learning, two essential issues are emerging.

C.3 *Optimization of Client-Server Communication:* Resources are commonly shared among IoT systems and services at the edge, especially the scarce or rated communication capabilities. Even though IFL transmits model parameters instead of raw data, iterative learning may still affect user experience (e.g., lagging for required contents). Hence, the client-server interaction shall be optimized to make IFL less burdensome [24], [36], [37].

C.4 *Aggregation of Local Updates:* To expedite the learning (i.e., reduce the number of iterations between the clients and the server) for an updated model reaching certain targets, i.e., predefined model accuracy, the heterogeneity hidden in local updates shall be properly addressed to not only accelerate the learning process but also improve model performance [26], [27], [38].

To overcome the emerging challenges in the periodic and iterative learning of IFL, several solutions are proposed.

C. Related Solutions

IL refers to the process of learning new knowledge continuously while tackling the forgetting of old knowledge. To achieve that, several methods to avert catastrophic forgetting in AI models are studied, e.g., weighted processing strategy [39], knowledge distillation [51], confrontation mechanism [52], etc. However, the dynamics of IFL, e.g., gradually increased non-IID data and diversified clients, may make them inefficient or even inoperative to assist various IoT systems and services. Such that, several solutions have been proposed by either utilizing a set of pertaining data that can preserve beneficial information for the later update [53] or controlling the forgetting paces of old classes locally at the client and globally at the server to incorporate the knowledge in emerging classes [16], [54], [55].

Even though current solutions can efficiently and effectively support structured IFL scenarios, e.g., to train AI models with sets of predefined incremental tasks, the time-varying and individual-dependent properties in real-world applications are yet discussed, especially for the joint optimization of periodic learning to determine when and how to start IFL with a proper set of clients selected, and iterative learning to save the client-server interaction cost and improve the overall training performance for an unbiased model.

1) *Solutions for Periodic Learning:* Instead of studying and assisting IFL in a periodic manner, current research tends to run incremental tasks either with a predefined data schema or in a continuous learning environment. Specifically, tasks are configured and executed by the learning participants with non-IID data settings to train a balanced and unbiased model through several local and global rounds [26], [30], [31], [32]. Moreover, the increase in data and clients are commonly tackled in one continuous learning task, lacking the thorough consideration of their impacts on related services to update their AI models rationally for a balance between invisible consumption of learning resources (e.g., local data, computation power, etc.) and visible Quality of Services (QoS) (e.g., service lagging, unexpected results, etc.) [16], [53], [54], [55].
   Since ubiquitous IoT consists of plentiful edge devices with redundant data and heterogeneous computing capabilities, before IFL starts periodically, a set of clients is selected to ensure that the learning can be cost-efficient. In general, there are two kinds of strategies for client selection, i.e., to be fixed or dynamic. As for the first kind, the server will cooperate with a set of clients consistently [40], [41], [42], [43]. Even though it can simplify the learning procedure, its strict restriction may lead to deficiencies, e.g., stragglers, overlearning, etc. Thus, dynamic approaches are discussed to either stochastically activating clients [26], [28], or filtering clients according to specific metrics, e.g., cumulative effectiveness during the learning [44], [45], time budget in local training [45], performance of local model [46], [47] and quality of local data [25], [47], [48].
2) *Solution for Iterative Learning:* During the interactions between the clients and the server, the main objective is to shorten the model training time (TT) with less CC. First, without hammering the overall model performance, intuitively, the optimization of client-server interaction can be made by reducing the uploading frequency of shared parameters [27], [43], as well as compressing the data packages to be exchanged [41], [45], [50]. More advanced, the overall communication efficiency can be further improved by adjusting the parallelism among clients to reduce the overall learning rounds [28], managing a hierarchical topology, e.g., with multiple servers to reduce network burden [40], and decreasing uploading ratios of model layers that make few contributions to the global model [25], [43], [56].
   Moreover, the heterogeneity among clients, e.g., non-IID data, asynchronous local training speed, etc., may retard the overall TT. Accordingly, as the step to alleviate the differences among clients, the model aggregation (to build the global model according to received local updates) is enhanced according to, e.g., a sample selection strategy to prioritize more informative data [49], a deadline-based mechanism to prevent the intolerable waiting for updates of particular clients [44], a proximal term measuring the contribution of each client to ensure the overall learning stability [26], and a weight factor to adjust the temporal and informative differences among local updates for fast convergence [25], [57].

In summary, as shown in Table I, first, current studies are mostly conducted under fixed scenarios without considering the dynamics of IoT systems and services. Second, different methods and mechanisms are proposed to address a specific topic in either client selection (C.2), communication optimization (C.3), or aggregation enhancement (C.4). Even though certain improvements can be achieved, the continuous changes in ubiquitous IoT systems and services, i.e., the gradually increased clients and their local data, are not well considered, which may make current solutions less efficient or even invalidated to address related challenges (C.1 to C.4). To fill the gap by addressing the four challenges in time-varying learning contexts, this article proposes SLMFed, which can accommodate the dynamics of IFL to not only remedy the impact of service growth but also maintain a model with a balanced performance on fresh and stale data.

III. Proposed SLMFed

To tackle the challenges encountered in IFL, a stage-based and layerwise mechanism, called SLMFed, is proposed. As shown in Fig. 2, it consists of two kinds of learning processes.

1) *Periodic Learning to Start the Stage Transition:* It determines the start of stage transition to alleviate the influences of the dynamics of IoT systems/services.
2) *Iterative Learning to Update the Global Model:* It updates the global model collaboratively with CCs reduced and model performance improved.

The two kinds of learning work jointly for a shareable and reusable model that can keep a balanced performance to support both old and new content.

A. Stage Transition in Periodic Learning

It is designed to support the dynamics in real-world applications through various learning stages. Specifically, in this study, the stage transition is defined as the following.

Definition of "Stage Transition": A stage represents a steady-state of an IoT system/service for a certain period of time. Within a stage, along with the growth of the service, the number of active users and the size of private data will increase gradually and simultaneously. However, on the contrary, the ability of pretrained AI models and the experience received by the users will decline either significantly (when the service is initially launched) or slightly (when the service becomes stable). Such a change is inevitable as the model encoded with already-seen knowledge may not properly handle situations unobserved, e.g., new classes in classification, new choices in recommendations, etc. As a result, an intolerable decrease in model performance and user experience will trigger the model update, and thus, lead the system/service to enter a new stage.

To determine a stage transition from the current stage S_i to a new stage S_{i+1}, the periodic learning is implemented to first monitor the service status, and then select QCs as the premise to start the iterative learning for the model update.

TABLE I
OVERVIEW OF RELATED WORKS (NOT SUPPORTED AND SUPPORTED)

Related Work	C.1	C.2	C.3	C.4	Highlights (+ pros and - cons)
FedProx [26]	○	●	○	●	A framework to tackle heterogeneity in federated learning: + A proximal term is used to measure the contribution of each client - Behindhand to the dynamics of IFL
FedDL [27]	●	○	●	○	An FL system for human activity recognition: + Dynamic layer sharing scheme is used to optimize client-server interaction - lacking the consideration of periodic learning of IFL
FedAvg [28]	○	●	●	○	The most classic synchronous FL aggregation algorithm: + A practical and communication-efficient method for FL - Mostly for static and simple scenarios
FCL-BL [39]	●	○	●	○	A federated continuous learning scheme based on broad learning: + Weighted processing strategy is used to solve the catastrophic forgetting problem - Concerns about training performance and accuracy
EdgeFed [40]	○	○	●	●	An optimization algorithm for FL based on edge computing: + A two-fold process for local updates is used to reduce communication and computational cost - Lacking the discussion of continuous data sensing
SoteriaFL [41]	○	○	●	○	A unified framework for private FL: + Local gradient estimators and shifted compression schemes are used to save communication costs - Lacking support for periodic learning and model aggregation
FedNTD [42]	●	○	●	○	A federated distillation algorithm: + Tackling forgetting issues based on the distillation method - The initialization of model update for the periodic learning is not addressed
FedLAMA [43]	○	○	●	●	A layer-wise model aggregation scheme for FL: + Finegrained aggregation strategy is introduced to improve the training performance - Challenges in periodic learning is not tackled
E3CS [44]	○	●	○	●	A stochastic client selection scheme for FL under a volatile context: + Exponential-weight selection algorithm is used to select participants - Less consideration in optimizing client-server communications
NOMA [45]	○	●	●	○	An adaptive federated learning with gradient compression: + Adaptive gradient quantization and sparsification is implemented to save the communication cost - Modest improvement in learning performance
REFL [46]	○	●	○	●	A reputation-enabled FL model aggregation method: + Reputation score is used for client selection and model aggregation - Without the consideration of saving communication costs
Oort [47]	○	●	○	○	An FL algorithm with participant selection: + Picking clients with high statistical and system utility - Excessive communication overhead when training large models
FedProf [48]	○	●	○	○	An algorithm to optimize FL over non-IID data: + Representation profile dissimilarity is used to select clients adaptively - Less efficient for the iterative learning of IFL
FedBalancer [49]	○	●	○	○	A framework for FL with sample selection: + Sample selection strategy is used to prioritize more informative data - Without consideration of client-server interaction and model aggregation
FedSI [50]	○	○	○	●	A federated continual learning method adapting the synaptic intelligence: + Finding a common parameter space for local models to mitigate model drifts - High communication overhead in learning the global model
CFedSI [50]	○	○	●	●	An upgraded FedSI with bidirectional compression and error compensation algorithm: + Bidirectional compression and error compensation algorithm ensuring communication efficiency - Poor performance in periodic learning
This paper (SLMFed)	●	●	●	●	A stage-based and layer-wise mechanism for IFL: + Dedicated indicators to determine the stage transition + Measuring information changes to activate qualified clients + Adaptive local model uploading to save communication costs + Layer-wise aggregation measuring the information richness to enhance the training performance

1) Service Status Monitoring: As shown in Fig. 2(a.1), under the assumption that the IoT service will expand its market share with a growth rate to continuously attract clients and gather data, current-in-use AI models may become decayed gradually, and need to be updated periodically. Such that, related indicators shall be defined and used to determine the start of stage transition. Given that in two adjacent stages, i.e., S_i and S_{i+1}, their accumulated data are D_i and D_{i+1}, which are the sum of private data owned by all the clients in S_i and S_{i+1}, respectively. It is worth noting that C_i the number of clients in S_i is different from C_{i+1}, as IoT devices may join or leave the service freely. Then, the incremental data $\hat{D}_{i\,:\,i+1}$ can be calculated according to

$$\hat{D}_{i\,:\,i+1} = D_{i+1} - (D_{i+1} \cap D_i). \tag{1}$$

According to $\hat{D}_{i\,:\,i+1}$, the percentage of clients with new data can be calculated according to (2), where $\hat{C}_{i+1}$ stands for the clients with new data, which is the output of a distinct function $f(*)$ to count the appearance of a client

$$\hat{P}_{i\,:\,i+1} = \frac{\hat{C}_{i+1}}{C_{i+1}} = \frac{f\left(\hat{D}_{i:i+1}\right)}{C_{i+1}}. \tag{2}$$

Since the new sensed data may not contain new knowledge, e.g., replicas of historical choices, $\hat{P}_{i\,:\,i+1}$ cannot be used

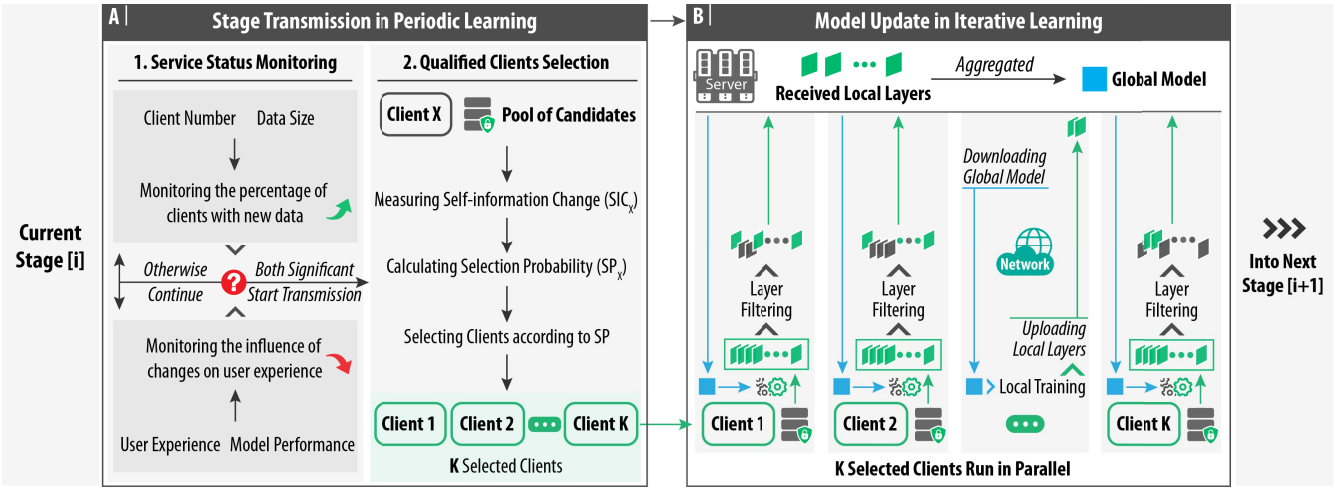


Fig. 2. Overall workflow of SLMFed. (a) Stage transition in periodic learning, which bridges stages to start the learning periodically. (b) Model update in iterative learning, which implements a layerwise process to update the global model.

directly as the only indicator to start the stage transition. Besides the changes in clients and data, a supplementary indicator to measure the changes in user experience shall be defined. Therefore, the average loss of w_i (which is the model generated in S_i) to process the data of $\hat{C}_{i+1}$ shall be considered, which is defined by

$$\hat{J}_{i\,:\,i+1,k} = L\big(d_{i\,:\,i+1,k}; w_i\big), \quad k \in \hat{C}_{i+1} \tag{3}$$

where $\hat{J}_{i\,:\,i+1,k}$ is the loss of w_i for a client k, $d_{i\,:\,i+1,k}$ is the local data of client k, and $L(*)$ is a user-specified loss function, which is task-oriented, e.g., cross-entropy loss for classification, and mean squared error for regression. Note that even though non-IID data owned by each client is harmful for training a unbiased model in FL, it can lessen the changes in $\hat{J}_{i\,:\,i+1,k}$, and in turn, alleviate the impact of information changes (i.e., catastrophic forgetting of old knowledge) to rapidly alter users' usual flavors by updating the model concerning more on new or special cases.

Furthermore, by comparing $\hat{J}_{i\,:\,i+1,k}$ with the model performance on the data set $d_{i,k}$ before the change happens in S_i, the general influence of changes on user experience $\hat{E}_{i\,:\,i+1}$ in IoT systems can be measured according to

$$\hat{E}_{i\,:\,i+1} = \left(\sum_{k}^{\hat{C}_{i+1}} E_k\right) / \hat{C}_{i+1}$$
$$\text{s.t.} \quad E_k = \begin{cases} 1, & \hat{J}_{i\,:\,i+1,k} \le L\big(d_{i,k}; w_i\big) \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Finally, by considering $\hat{P}_{i\,:\,i+1}$ and $\hat{E}_{i\,:\,i+1}$ jointly, a rational decision can be made to trigger the stage transition. Because $\hat{P}_{i\,:\,i+1}$ can highlight the proportion of changes in clients and data, and $\hat{E}_{i\,:\,i+1}$ can measure the actual impact of changes on model performance and user experience.

2) Qualified Client Selection: Before the stage transition actually starts to update AI models, QCs shall be selected from a pool of candidates that can participate in the learning, as it will be inoperable and costly to have all candidates activated for the model update. Moreover, it is also impactful with all candidates activated, as the model will be overleant on abundant and biased data samples. To tackle that, as shown in Fig. 2(a.2), the client is selected based on an activation probability that represents the level of self-information changes.

Specifically, based on relative entropy or Kullback–Leibler Divergence, a self-information change indicator (SIC) for each candidate, denoted as $\text{SIC}_{i\,:\,i+1,k}$, can be calculated according to (5), where $DV(*)$ is the divergence function; $d_{i,k}$ and $d_{i+1,k}$ are the two data sets of a client k in S_i and S_{i+1}, respectively; l_n is the total number of labels; and c is a constant used to avoid the generation of infinite value

$$\begin{aligned} \text{SIC}_{i\,:\,i+1,k} &= DV\big(d_{i+1,k} || d_{i,k}\big) \\ &= \sum_{l=1}^{l_n} \left(p_{l,i+1} \times \frac{\log_2(p_{l,i+1}+c)}{\log_2(p_{l,i}+c)}\right). \end{aligned} \tag{5}$$

In general, a client with a higher SIC indicates that its newly sensed data is more different from the old one. Since the performance dropping of the current model is caused by the new information, clients with higher values of SIC shall have a higher chance to be selected. Hence, the selection probability $SP_{i\,:\,i+1,k}$ can be calculated by using a Softmax function with $\text{SIC}_{i\,:\,i+1,k}$ and the data size of the client $N_{i+1,k}$ as the input

$$SP_{i\,:\,i+1,k} = \text{Softmax}\big(\text{SIC}_{i\,:\,i+1,k}, N_{i+1,k}\big). \tag{6}$$

Finally, a set of QCs can be activated through a random selection function as defined in (7), where $SP_{i\,:\,i+1}$ is the set of $SP_{i\,:\,i+1,k}$, and α as a hyperparameter controls the number of clients to be selected. Note that X the total number of clients in the pool is equal to C_{i+1}

$$\text{QC} = \text{Random}(SP_{i\,:\,i+1} \mid \alpha)$$
$$\text{s.t.} \quad SP_{i\,:\,i+1} = \{SP_{i\,:\,i+1,1}, SP_{i\,:\,i+1,2}, \ldots, SP_{i\,:\,i+1,X}\}. \tag{7}$$

In summary, such a client selection process is beneficial, as the randomness can keep a balance between the old and new data to avoid the overleaning issue for a less biased model. Its efficiency is tested in Section IV-A5.

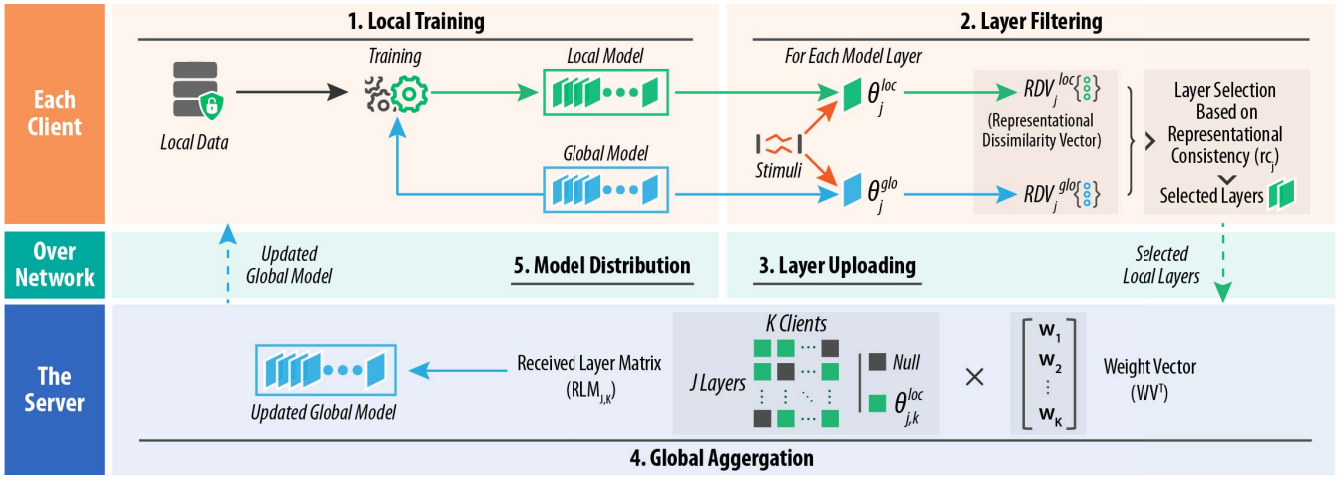


Fig. 3. Overall workflow of layerwise model update for IFL. Note that the flow ends when predefined termination conditions are met, i.e., target loss, maximum iterations, etc.

B. Model Update in Iterative Learning

As shown in Fig. 2(b), after a stage transition is triggered, the iterative learning starts to update the AI model. Since IFL can be intrusive for the edges running multiple applications simultaneously, it is critical to orchestrating QCs in learning a shareable global model from w_i in S_i to w_{i+1} in S_{i+1} cost-effectively.

Hence, a layerwise model uploading and aggregation process is designed and implemented in SLMFed, which consists of five consecutive phases as shown in Fig. 3.

1) Phase 1 Local Training: After the selection of QCs, the local training in each client can be started. Since the learning in S_{i+1} is iterative (until the model gets converged or the maximum number of iterations is reached), within a learning iteration t, a client k will train a local model $w_{t,k}^{\text{loc}}$ according to (8), where w_{t-1}^{glo} is the current global model in use, which is generated and updated in the previous iteration $t-1$; $d_{i+1,k}$ is the local data of client k in stage $i+1$; η is the learning rate; and $J_k'(\cdot)$ is the local gradient

$$w_{t,k}^{\text{loc}} = w_{t-1}^{\text{glo}} - \eta J_k'\left(d_{i+1,k}; w_{t-1}^{\text{glo}}\right). \tag{8}$$

2) Phase 2 Layer Filtering: Conventionally, once the local model is trained, it will be updated to the server directly. However, since AI models, i.e., DNNs, consist of multiple layers as defined in (9) (where θ_j is the parameter of the j^{th} layer, and J is the total number of model layers), and the importance of layers may vary from each other, e.g., the difference between shallow and deep layers, it is rational to upload layers adaptively to save CC. Hence, a layer filter is applied to select layers to be uploaded according to their RCs

$$w_{t,k}^{\text{loc}} = \left\{\theta_{j,k}^{\text{loc}},\ j \in J\right\}. \tag{9}$$

Moreover, as for the calculation of RC, a representational dissimilarity vector (RDV) is sampled from the conventional representational dissimilarity matrix to store the results of a layer j before and after the local training, noted as θ_j^{glo} and θ_j^{loc}, in processing pairs of common stimuli. Accordingly, the rc_j of the j^{th} layer can be calculated according to (10), where $\text{RDV}_j^{\text{glo}}$ and $\text{RDV}_j^{\text{loc}}$ are elements in related RDVs, respectively, and ρ is the Pearson correlation coefficient

$$rc_j = \rho\left(\text{RDV}_j^{\text{glo}}, \text{RDV}_j^{\text{loc}}\right)^2. \tag{10}$$

Finally, based on the RCs of local layers, corresponding uploading possibilities *UPs* can be calculated according to (11). Then, an adaptive decision can be made based on $UP_{k,j}$. Note that if the value of UP is higher, the corresponding layer will have a higher chance to be uploaded, and otherwise not

$$\begin{aligned} UP_k &= \text{Softmax}(rc_k) \\ &\text{s.t.}\quad rc_k = \{rc_{k,1}, rc_{k,2}, \ldots, rc_{k,J}\}. \end{aligned} \tag{11}$$

3) Phase 3 Layer Uploading: When the local layers are filtered, they will be grouped and uploaded to the server through a private or public network (i.e., Intranet or Internet). Nevertheless, since the network is vulnerable to sniffing and poisoning attacks, compression and encryption techniques can be applied to ensure the stability, reliability, and security of data transfers.

4) Phase 4 Global Aggregation: Instead of aggregating local models directly, the server needs to update the global model according to the local layers received from QCs. To achieve that, first, a received layer matrix (RLM) is defined in (12). Note that RLM has some $\theta_{j,k}$ with NULL values, as layers are selectively uploaded

$$\text{RLM}_{J,K} = \begin{bmatrix} \theta_{1,1}^{\text{loc}} & \theta_{1,2}^{\text{loc}} & \cdots & \theta_{1,K}^{\text{loc}} \\ \theta_{2,1}^{\text{loc}} & \theta_{2,2}^{\text{loc}} & \cdots & \theta_{2,K}^{\text{loc}} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{J,1}^{\text{loc}} & \theta_{J,2}^{\text{loc}} & \cdots & \theta_{J,K}^{\text{loc}} \end{bmatrix}. \tag{12}$$

Furthermore, since the local layers are trained based on non-IID data, to steer the learning direction and avoid the overlearning issue, a weight vector measuring the IR of local data can be created according to (13), where $WV \in \mathbb{R}^K$, N_k

Algorithm 1 SLMFed for Each Client in Parallel

PART 1: Running Status Monitoring in each client
Report to the server if new data are sensed
Report to the server if performance dropping is observed
Wait for the stage transition decision from the server
if Stage transition is "TRUE" **then**
    Transmit its SIC (defined in Formula (5)) to the server
    Wait for the signal of client selection
end if
PART 2: Iterative Learning in the Selected Client
if The client is "SELECTED" **then**
    Train its local model w^{loc} according to Formula (8)
    Select layers according to *UP* (Formula (11))
    Calculate its *IR* (Formula (13))
    Transmit w^{loc} and *IR* to the server
    Receive the training command from the server
    if Termination signal "STOP" is received **then**
        Stop the learning and deploy the global model
    end if
end if

Algorithm 2 SLMFed for the Server

PART 1: Stage Transition Determination
Receive heartbeat signals from clients
Calculate stage transition indicators
if Stage transition is required **then**
    Return "TRUE" to clients
end if
if Stage transition is "TRUE" **then**
    Receive SIC and make client selection (Formula (7))
    Send "SELECTED" to qualified clients (QCs)
end if
PART 2: Iterative Learning for Global Model Update
Receive local updates from QCs
Cacluate WV (Formula (13))
Update global model (Formula (14))
if termination condition is met **then**
    Send "STOP" and updated global model to the clients
else
    Send "CONTINUE" and updated global model to QCs
end if

is the size of data used in client k, and ir_k is the information entropy or label number of local data of client k

$$WV = \{w_1, w_2, \ldots, w_K\} \quad \text{s.t.} \quad w_k = \text{Softmax}(ir_k, N_k). \tag{13}$$

Finally, by applying *WV* on *RLM* in the learning iteration t, the global model w_t^{glo} can be updated according to a layerwise model aggregation function as defined in

$$w_t^{\text{glo}} = \text{RLM}_t \times WV_t^T. \tag{14}$$

5) Phase 5 Model Distribution: Through the downstream of the connection between the server and clients, the updated global model (i.e., w_t^{glo}) is distributed to the clients to either start a new learning iteration or stop the learning with the global model deployed on all the clients.

In summary, the above procedure can not only significantly reduce the CC but also dramatically improve the model performance, which is evaluated in Section IV-B.

C. Algorithm of SLMFed

As SLMFed involves interactions between the clients and the server, it needs to be deployed on both sides.

1) *SLMFed for Each Client:* As shown in Algorithm 1, it consists of two parts, namely, running status monitoring in each client and iterative learning in the selected client. Specifically, in the first part, the client will send its heartbeats to the server when changes in sensed data and service performance are detected. While the state transition decision is made by the server and QCs are identified, in the second part, the selected client enters the iterative learning for local training, layers selection, and local parameters uploading. Through the interaction with the server, the global model will be updated in a collaborative and privacy-preserving manner.

2) *SLMFed for the Server:* As shown in Algorithm 2, it also consists of two parts, namely, stage transition determination and iterative learning for the global model update. Specifically, in the first part, according to the status report from the clients, the server will calculate related indicators and make the decision to start the stage transition with QCs selected. Moreover, in the second part, the server will orchestrate QCs and aggregate local layers received from them for the global model. After that, the server will check if the stop condition (i.e., maximum iterations) is met and return the checking result together with the updated global model to the client.

In summary, SLMFed consists of periodic and iterative learnings to implement a cost-efficient and performance-stabilized IFL. In particular, by measuring changes in client number, sensed data, model performance, and user experience, it can make rational decisions on when to trigger stage transition and how to select QCs. Moreover, within iterative learning, a layerwise model uploading and aggregation process is implemented to update the global model with balanced performance on both stale and fresh data.

IV. Performance Evaluation

In this section, the performance of SLMFed is evaluated and discussed. First, common settings are introduced. Next, SLMFed is compared with state-of-the-art baselines to demonstrate its supremacy in supporting IFL.

A. Common Settings

For fairness, common settings for evaluation tasks, scenarios, methods, metrics, and stages are given.

1) Evaluation Tasks: Four common tasks based on four standard data sets are defined to learn corresponding

convolution neural networks (CNNs), respectively. Specifically, the configuration for each task is described below.

1) *Task 1 With MNIST*[1] *Data Set:* It contains 60 thousand training samples and 10 thousand testing samples in 10 labels, and its image size is $28 \times 28 \times 1$. Within this task, each client possesses 200-600 training samples with 1-6 classes, and the CNN model is designed with 2 convolutional layers and 2 fully connected layers.
2) *Task 2 With FMNIST*[2] *Data Set:* Similar to MNIST, it contains 60 thousand training samples and 10 thousand testing samples in 10 labels, and its image size is $28 \times 28 \times 1$. Within this task, each client possesses 200-600 training samples with 1-6 classes, and the CNN model to be learned shares the same structure of MNIST.
3) *Task 3 With CIFAR-10*[3] *Data Set:* It contains 50 thousand training samples and 10 thousand testing samples in 10 labels, and its image size is $32 \times 32 \times 3$. Within this task, each client possesses 165-500 training samples with 1-6 classes, and the CNN model is designed with 3 convolutional layers and 2 fully connected layers. Specifically, due to the complexity of CIFAR-10, batch normalization modules are used to improve training efficiency.
4) *Task 4 With GTSRB*[4] *Data Set:* It contains 34 799 training samples and 12 630 testing samples with 43 labels, and its image size is $32 \times 32 \times 3$. Within this task, each client possesses 116-348 training samples with 1-22 classes, and the CNN model is designed with 4 convolutional layers and 2 fully connected layers. Similar to CIFAR-10, batch normalization modules are also used in this task.

Notably, considering the difficulties in learning corresponding CNNs, the abovementioned tasks can be divided into two groups, namely, simple tasks, including MNIST and FMNIST, and complicated tasks including CIFAR-10 and GTSRB. In general, such a setting can better evaluate the generalizability of SLMFed to support IFL in various IoT systems and services.

2) Evaluation Scenarios: To mimic real-world situations, where IFL is applied, an IoT system is simulated with the following.

1) *IoT Devices:* In each task, 200 devices are visualized with non-IID data generated from the corresponding standard data set. They act as clients in IFL to continuously sense data, update models, and exchange parameters.
2) *IoT Network:* It links the server and clients in IFL. Note that a random delay of 10s to 100s is configured while clients are communicating with the server to imitate the communication latency.

Based on the system, two incremental settings are designed.

1) *Client Growth:* For each task, initially, 100 clients (out of 200) are activated, and the number of available clients will increase by 1% to 1.5% per learning round until the total number (i.e., 200) is reached. It reflects the system gradually penetrates the market to attract more users.
2) *Data Growth:* For each client, its data will grow gradually to simulate the data accumulation process during the usage of the system. Specifically, a client will have a 50% chance to increase its local data with two predefined data incremental ranges, i.e., 1) 0% to 1%, and 2) 0% to 10%. Note that new or duplicated samples may be assigned to clients, and such a setting is used, as, in reality, the data increase timing and rate for each user can be different.

Note that as two data incremental ranges are used for each task, there are, in total, 8 evaluation cases to test the robustness of SLMFed in supporting IFL in various dynamic IoT contexts.

3) Evaluation Methods: Thirteen methods are used.

1) *FedAvg [28]:* It is the classic synchronous FL algorithm using an average function for model aggregation.
2) *FedProx [26]:* It addresses the heterogeneity among devices and data by using tolerating partial work and proximal terms, whose hyperparameter μ is set to 1.
3) *SoteriaFL [41]:* It optimizes client-server communication by transmitting encrypted and compressed parameters. The encryption factor ξ conforms to normal distribution.
4) *FedNTD [42]:* It enhances the local training process of clients to tackle knowledge forgetting. Its hyperparameters τ and β are set to 3 and 1, respectively.
5) *FedLAMA [43]:* It adjusts the aggregation interval in a layerwise manner. The interval increase factor ϕ is 1.2.
6) *E3CS [44]:* It randomly selects clients based on an exponential-weight algorithm.
7) *NOMA [45]:* It selects clients based on a fairness index. Its hyperparameters σ^2, τ, and p_k are set to -174, 1, and 0.1, respectively.
8) *Oort [47]:* It selects clients by the training utility and the quality of local data.
9) *FedProf [48]:* It selects clients by considering a representation profile dissimilarity.
10) *FedBalancer [49]:* It adopts a sample selection strategy to speed up global training.
11) *FedSI [50]:* It implements federated continual learning by adapting the synaptic intelligence. Its hyperparameter ξ is set to 1.
12) *CFedSI [50]:* It improves FedSI with the bidirectional compression and error compensation algorithm to ensure communication efficiency and training convergence.
13) *SLMFed:* It is the proposed method with dedicated strategies to support both periodic and iterative learning of IFL.

4) Evaluation Metrics: To comprehensively reveal the performance, two kinds of metrics, i.e., to be general or stage-specific, are defined. First, the general metrics include model accuracy, TT, and CC.

1) *Accuracy (AC):* It is calculated by (15), where *TP*, *TN*, *FP*, and *FN* represent true positives, true negatives, false positives, and false negatives, respectively.

[1] http://yann.lecun.com/exdb/mnist
[2] https://github.com/zalandoresearch/fashion-mnist
[3] https://www.cs.toronto.edu/~kriz/cifar.html
[4] https://bitbucket.org/jadslim/german-traffic-signs

2) *TT:* It is calculated by (16), where N_r is the round number, and $T_{s,r}$ and $T_{k,r}$ are the TT of the server and client k in round r, respectively. Since all the clients in IFL run in parallel, the maximum $T_{k,r}$ is summed up with $T_{s,r}$ to measure the time of client spent in each learning round.
3) *CC:* As defined in (17), it consists of $\mathrm{CC_{up}}$ and $\mathrm{CC_{down}}$ generated when the clients upload local parameters to the server, and the server distributes updated global parameters to the clients, respectively. According to [25], the cost is measured by the size of parameters transmitted over the network.

$$\mathrm{AC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}} \tag{15}$$

$$\mathrm{TT} = \sum_{r=1}^{N_r}\left(T_{s,r} + \mathrm{Max}(T_{k,r})\right). \tag{16}$$

$$\mathrm{CC} = \sum_{r=1}^{N_r}\left(\mathrm{CC_{up}} + \mathrm{CC_{down}}\right). \tag{17}$$

Furthermore, three stage-specific metrics are defined to analyze the performance across stages.

1) *Maximum AC (MAC):* As defined in (18), MAC_i is the MAC achieved in Stage i.
2) *Stability of AC (SAC):* It is calculated by (19), where μ_i and σ_i represent the mean and standard deviation of AC in Stage i, respectively. Note that a higher SAC indicates the learning fluctuates less with a similar AC around the mean value per learning round.
3) *Stage Contribution in AC (SCAC):* Based on the normalized variation of AC at the beginning and end of a single stage, SCAC calculates the contribution of a method within a specified stage via (20), where ς is a constant (by default 1.22) to avoid negative values.

$$\mathrm{MAC}_i = \mathrm{Max}(\mathrm{AC}_i). \tag{18}$$

$$\mathrm{SAC}_i = \frac{\mu_i}{e^{\sigma_i}}. \tag{19}$$

$$\mathrm{SCAC}_i = \tan\frac{\mathrm{AC}_{i,\mathrm{end}} - \mathrm{AC}_{i,\mathrm{begin}}}{100} + \varsigma. \tag{20}$$

In general, the method with higher MAC, SAC, and SCAC is superior, as it can learn a more competitive model that can not only achieve high accuracy but also maintain balanced performance on both old and new data.

5) Evaluation Stages: To study IFL in a dynamic environment, where clients and data are growing gradually, the first five stages in periodic learning are studied in the evaluation. To control the stage transition, as described in Section III-A, two indicators, i.e., P the percentage of clients with new data, and E user received experience (directly related to model performance, how correct the model is to make the image classification), are used. Accordingly, how to choose an appropriate value as the threshold to trigger the stage transition efficiently and effectively needs to be addressed.

To resolve that, by adopting the idea of the Monte Carlo method, the impact of P and E on the model performance is pretested. As shown in Fig. 4, in Step 1, the relationships among AC, P and E are analyzed, and then, in Step 2, the changes of AC, noted as ACCs, are calculated to measure AC differences between two adjacent learning rounds. After that, in Step 3, based on ACCs, the turning points (with higher ACC values) can be identified, and finally, in Step 4, the distributions of P and E bringing impacts on model learning can be drawn. Following the above steps, the pretest can be executed several times to find a stable and fine distribution of P and E, based on which, the P and E binging more significant impacts to train the models for the four tasks (i.e., MNIST, FMNIST, CIFAR-10, and GTSRB) can be defined, namely, 41%, 39.8%, 40%, and 40% for P, and 75%, 65%, 55%, and 70% for E. Note that the transition condition is bigger than P and smaller than E, as new data and user experience in each client will increase and decrease, respectively.

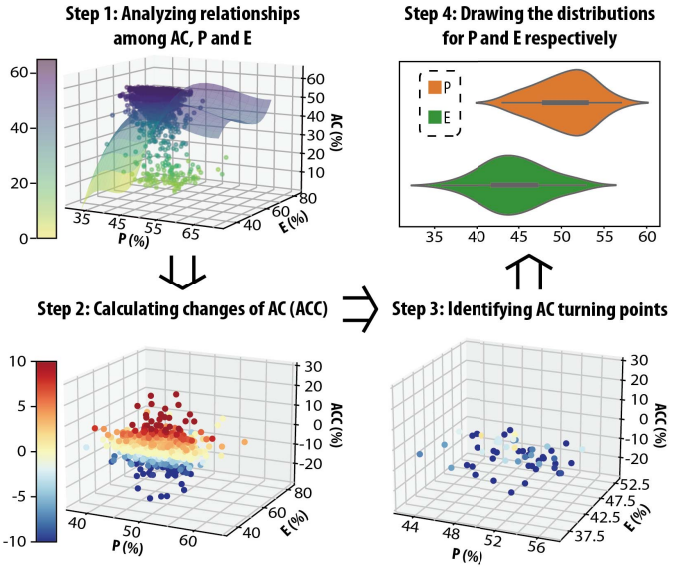


Fig. 4. Steps to determine stage transition indicators, i.e., P and E.

Moreover, before the iterative learning starts in a stage, QCs need to be selected for the global model update. As described in Section III-A, a random client selection process with SP selection probability (which is calculated based on the self-information change indicator) is used. Accordingly, the performance of three mechanisms, namely, random selection without SP (in which, each client has the same probability to be selected), ranked selection with SP (in which, clients with higher values of SP are selected), and random selection with SP (in which, clients are randomly selected based on SP), are pretested. As shown in Fig. 5 (where ACCs between stages are presented), random selection with SP achieves better performance than others for 4 training tasks in 2 incremental scenarios, as the randomness guided by the information changes can be beneficial for IFL to learn a model with knowledge from both old and new data. Therefore, it is used by default in the evaluation. Note that to simplify the pretesting, FedAvg is used as the default method to train the model for each task.

In summary, during the experiment, five learning stages are defined. Specifically, between stages, the transition is controlled by the predefined P and E, and within a stage, the

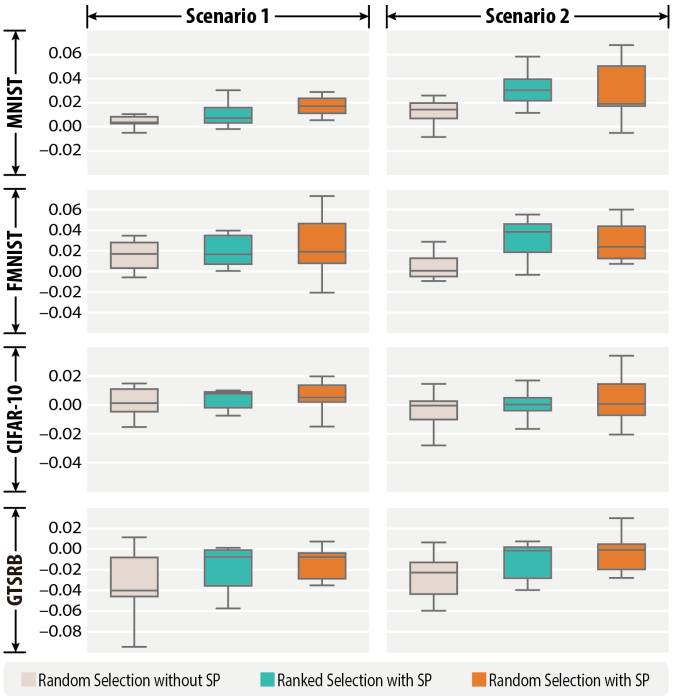


Fig. 5. Boxplot of ACCs between stages for three client selection mechanisms.

random client selection with SP is used to activate 25% of the available clients for the model training. In general, while running the learning according to the above settings, each task can be treated as a set of five subtasks, each of which is equivalent to a stage. Hence, in each sub-task/stage, when a predefined stop condition, e.g., the maximum number of rounds or minimum loss, is reached, the subtask ends, and it waits for the stage transition to start the next. During the waiting, the data and clients will increase incrementally, in which, the maximum waiting time defined for the learning round is used as the timer to trigger the increase of data and clients, and also the checking of stage transition condition. It is worth noting that to reduce the complexity of the learning, the number of QCs and the amount of local data used in a stage will remain unchanged, and their newly accumulated data during the learning will be used in the following stage.

B. Evaluation Results

Based on the above settings, the evaluation is conducted to reveal the performance of compared methods within and across stages.

1) Performance Within Stages: First, as listed in Tables II and III, compared to the twelve baselines, SLMFed can achieve the highest accuracy and stabilize the learning as indicated by the values of MAC and SAC, respectively, in almost all the stages that are specified by the tasks (i.e., MNIST, FMNIST, CIFAR-10, and GTSRB). Specifically, while comparing MAC of SLMFed with the ones of the baselines in the final stage of each task, the average performance boost in the two incremental scenarios is about 8.97%, 17.69%, 35.65%, and 66.05% for MNIST, FMNIST, CIFAR-10, and GTSRB, respectively. It shows that the accumulated information, regardless it is new or old, can be better handled by SLMFed to learn high-performance and stable models.

Moreover, while diving into each task, as illustrated by AC curves of the five consecutive stages in Fig. 6(a) and (b), it can be observed that for simple tasks (i.e., MNIST and FMNIST), SLMFed can consistently outperform other baselines with less TT to get the models converge. It shows that through the corporation of the layerwise model aggregation, the adaptive local model uploading strategy, which reduces the uploading frequency of local model layers with similar RCs, will not affect the overall model performance but, surprisingly, can be beneficial to avoid the overlearning on common or biased knowledge extracted from non-IID data preserved by the clients.

Finally, even though SLMFed can overcome the twelve baselines in simple tasks at the early stage, it is tied with them for the complex tasks (i.e., CIFAR-10, and GTSRB) in the first two stages, as illustrated in Fig. 6(c) and (d). It is because, during the evaluation, the amount of data and the number of clients are gradually increasing, and a scarcity of training samples is experienced, making the method underperformed. Once local data becomes plentiful, as illustrated by the curves at the last three stages in Fig. 6(c) and (d), the merit of SLMFed is disclosed, since it can gain the potential to further elevate model performance when other baselines become incapable of avoiding the forgetting on old knowledge as well as absorbing the new one.

2) Performance Across Stages: The stage-based learning in the evaluation can be treated as an entire learning process to reveal the overall performance of SLMFed. First, as shown in Fig. 7, regardless of the complexity of the learning tasks, a significant improvement can be achieved by SLMFed under a specified CC for each task (i.e., 6G, 17G, 35G, and 50G for MNIST, FMNIST, CIFAR-10, and GTSRB, respectively) to maintain the highest AC. Compared to other methods in each task, it can improve AC by about 150.73%, 142.87%, 34.02%, and 96.15%, respectively, and then in each scenario, it can improve AC by about 146.80% and 65.08%, respectively.

Moreover, by analyzing the results in Tables II and III, and Fig. 8 (which shows the accumulated SCAC across stages) jointly, it can be seen that SLMFed can achieve a high-AC growth at the first stage in almost all the cases (i.e., totally $8 = 2$ scenarios $\times$ 4 tasks), and then remain a stable growth throughout the rest of the stages in each case. Numerically, about 17.96% increase in SCAC can be observed in all cases for SLMFed. Such a result shows the supremacy of SLMFed in periodic learning and iterative learning.

1) *Avoiding Catastrophic Forgetting in Periodic Learning:* In the first stage, AC of all the compared methods can climb up rapidly, and then grow gradually in the following stages along with the increase of the clients and local data. Moreover, all the methods do not experience a significant dropping in overall model performance, while new data are consistently accumulated and used to update the global model. It shows that a proper learning context for IFL can be controlled and prepared by the proposed stage transition and client selection strategies in SLMFed.

TABLE II
VALUES OF MAC AND SAC AT EACH STAGE IN DIFFERENT SCENARIOS FOR SIMPLE TASKS (INCLUDING MNIST AND FMNIST)

Task	Scenario	Method	Stage 1		Stage 2		Stage 3		Stage 4		Stage 5	
			MAC↑	SAC↑	MAC↑	SAC↑	MAC↑	SAC↑	MAC↑	SAC↑	MAC↑	SAC↑
MNIST	1	SLMFed	**88.51***	**0.6037**	**92.92**	0.901†	**94.54**	**0.9313**	**95.65**	**0.949**	**96.33**	**0.958**
		FedAvg	87.31	0.5384	91.34	0.8871	92.72	0.9085	95.09	0.9415	96.17	0.956
		FedProx	17.53	0.1111	39.88	0.3413	55.82	0.5065	68.33	0.5888	68.43	0.6508
		NOMA	87.31	0.5378	91.34	0.8871	92.99	0.9075	95.09	0.9412	96.17	0.9552
		FedProf	74.91	0.5435	77.36	0.7563	78.59	0.7765	84.73	0.7846	86.84	0.8603
		FedLAMA	83.47	0.4389	87.6	0.848	89.43	0.8785	91.44	0.9016	92.48	0.9191
		FedNTD	87.65	0.566†	91.64	0.8889	93.09	0.9111	95.33	0.9447	96.19†	0.9575†
		SoteriaFL	87.79†	0.5464	91.43	0.8893	93.43	0.919†	95.08	0.9422	96.01	0.9554
		Oort	77.87	0.5427	90.72	0.8463	93.48†	0.9162	94.83	0.937	94.82	0.9445
		E3CS	83.39	0.429	89.08	0.8647	90.57	0.8958	91.96	0.9124	92.75	0.925
		FedBalancer	87.31	0.5384	92.89†	**0.9016**	93.16	0.9172	95.39†	0.947†	95.26	0.9499
		FedSI	26.82	0.1369	63.85	0.4906	72.92	0.6834	74.71	0.7363	76.39	0.7544
		CFedSI	24.71	0.1159	67.17	0.497	72.57	0.6952	81.82	0.7514	84.24	0.8314
	2	SLMFed	**88.51**	**0.5949**	**93.68**	**0.9031**	**96.32**	**0.9499**	**97.3**	**0.9655**	**97.93**	**0.9756**
		FedAvg	87.31	0.5378	92.7	0.8928	94.87	0.9202	97.07	0.9625	97.87	0.9744
		FedProx	17.53	0.1111	39.28	0.3537	59.95	0.5534	71.18	0.67	64.48	0.5921
		NOMA	87.31	0.5384	92.7	0.8928	94.93	0.9203	97.22†	0.9629	97.89	0.9753
		FedProf	74.91	0.5504	78.82	0.7657	93.13	0.8449	91.99	0.9093	90.97	0.9037
		FedLAMA	83.47	0.4389	89.25	0.8577	91.88	0.8989	94.18	0.9283	95.76	0.9505
		FedNTD	87.65	0.5641†	92.86	0.8965	95.08	0.9251	97.19	0.964†	97.92†	0.9753†
		SoteriaFL	87.79†	0.5464	92.92†	0.8973	95.49	0.9352	97.1	0.9628	97.77	0.9741
		Oort	77.87	0.544	92.21	0.8601	95.81†	0.9383†	96.97	0.9629	97.17	0.9694
		E3CS	83.39	0.429	89.08	0.8647	90.57	0.8958	91.96	0.9124	92.87	0.9253
		FedBalancer	87.31	0.5384	92.92	0.9018†	93.16	0.9172	95.39	0.947	95.26	0.9501
		FedSI	26.82	0.1369	79.77	0.5764	84.53	0.8142	94.09	0.8835	95.84	0.9494
		CFedSI	24.71	0.1159	79.04	0.5613	83.17	0.8008	93.58	0.8608	95.61	0.9435
FMNIST	1	SLMFed	**61.72**	**0.4437**	**75.32**	**0.6921**	**78.34**	**0.7578**	**79.42**	**0.7771**	**80.86**	**0.7867**
		FedAvg	61.61†	0.4309	71.25	0.6419	76.07†	0.7302	76.27	0.7524	77.86	0.7713
		FedProx	10.7	0.107	13.27	0.1132	41.96	0.3861	33.03	0.2907	40.39	0.3826
		NOMA	61.61†	0.4306	71.25	0.6419	75.79	0.7284	76.29	0.751	77.44	0.7657
		FedProf	54.36	0.3882	58.78	0.5649	59.15	0.5741	60.21	0.5892	78.17	0.6655
		FedLAMA	52.2	0.3036	55.03	0.5197	57.93	0.5619	68.36	0.6316	68.73	0.6821
		FedNTD	61.5	0.4419†	71.88	0.6763†	76.03	0.7371†	76.92	0.7549	78.68†	0.7749†
		SoteriaFL	61.33	0.4164	70.93	0.6425	76.06	0.7336	75.77	0.7449	77.47	0.7657
		Oort	60.66	0.423	75.31†	0.6541	75.3	0.737	77.28†	0.7615†	75.92	0.736
		E3CS	55.49	0.3484	62.4	0.597	70.12	0.6367	72.96	0.7136	74.85	0.7418
		FedBalancer	61.35	0.4312	72.45	0.6475	73.59	0.7158	76.58	0.7475	76.81	0.7534
		FedSI	11.11	0.1111	35.16	0.2024	49.89	0.3261	65.45	0.5538	73.63	0.6831
		CFedSI	11.11	0.1111	27.27	0.2175	47.77	0.3513	65.3	0.5641	66.27	0.6374
	2	SLMFed	**61.72**	**0.4452**	**79.11**	0.7083†	**83.95**	**0.793**	**86.11**	**0.8418**	**87.31**	**0.8612**
		FedAvg	61.61†	0.4304	73.57	0.6784	80.21	0.7635	83.69	0.809	84.98	0.8418
		FedProx	10.7	0.107	19.09	0.1381	42.81	0.4087	31.61	0.3119	45.87	0.4159
		NOMA	61.61†	0.4304	73.57	0.6784	79.73	0.7591	83.5	0.8073	84.77	0.8393
		FedProf	54.36	0.3882	70.13	0.5777	80.85†	0.6937	84.06	0.8268†	87.04†	0.8398
		FedLAMA	52.2	0.3036	70.33	0.5935	78.51	0.7411	79.18	0.7646	81.39	0.8077
		FedNTD	61.5	0.4415†	73.96	0.688	80.55	0.7725†	84.08†	0.8193	85.63	0.8483†
		SoteriaFL	61.33	0.4155	73.32	0.671	80.36	0.768	83.4	0.7986	84.8	0.8394
		Oort	60.66	0.423	77.77†	**0.7136**	80.19	0.7608	83.59	0.8119	86.8	0.8386
		E3CS	55.49	0.3484	62.4	0.597	70.12	0.6367	72.96	0.7136	74.93	0.7425
		FedBalancer	61.35	0.4312	72.45	0.6476	73.59	0.7158	76.58	0.7475	76.81	0.7544
		FedSI	11.11	0.1111	41.66	0.2448	71.62	0.5749	74.53	0.7243	76.32	0.7513
		CFedSI	11.11	0.1111	21.33	0.1095	69.78	0.4164	77.04	0.7141	78.1	0.7611

* Bolded text indicates the best value among the compared methods in a stage.
† Underlined text indicates the second best value among the compared methods in a stage.

2) *Boosting Model Performance in Iterative Learning:* Compared to the baselines, except that SLMFed is less competitive at the first two stages in the second incremental scenario of CIFAR-10 (according to Tables II and III), it can achieve a higher AC across the stages in rest cases. Moreover, as illustrated

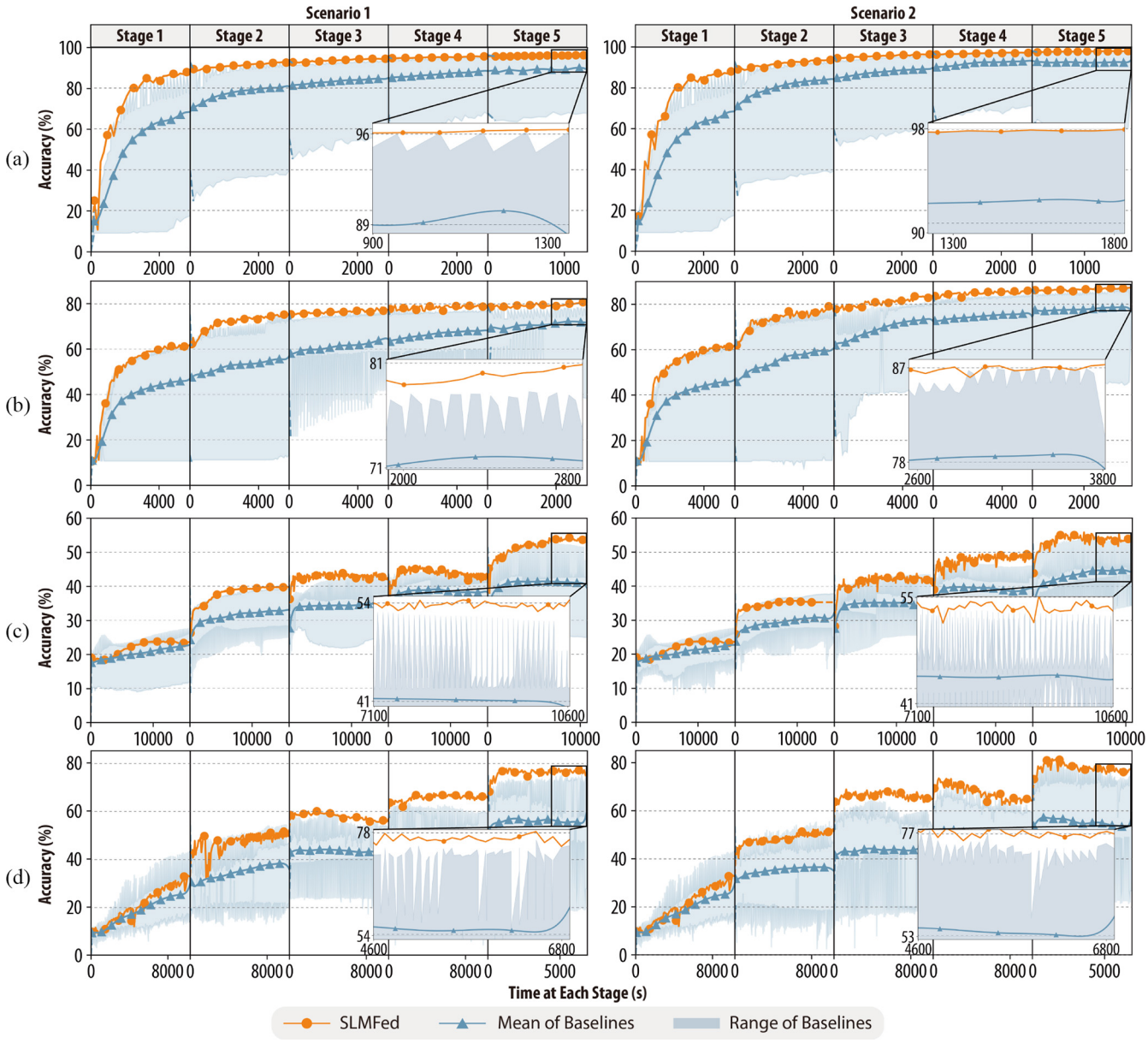


Fig. 6. AC curves of the proposed SLMFed, along with mean distribution and overall range of other twelve compared baselines within each stage for four data sets in two client and data incremental scenarios. (a) MNIST. (b)FMNIST. (c) CIFAR-10. (d) GTSRB.

by the gradually increased SCAC across stages for SLMFed in Fig. 8 (except the case in scenario 2 of GTSRB that SCAC drops in the fourth stage), such an improvement is also more consistent and robust than the twelve baselines. It shows that the layerwise optimization implemented by SLMFed can better steer the learning direction of IFL in the dynamic context by adaptively extracting and aggregating the informative parameters learned from QCs.

In summary, compared to the baselines, SLMFed can better support IFL in dynamic IoT contexts simulated by the two incremental scenarios for the four tasks. In general, the stage-based periodic learning and layerwise iterative learning of SLMFed can tackle the issue of performance dropping in handling old and new data, and resolve the issue of overlearning on redundant information, respectively. Therefore, it can not only maintain outstanding performance but also significantly reduce the learning time and cost in each IFL stage.

C. Discussions

Through the above evaluation, the following observations about SLMFed can be found.

1) It is robust to support IFL in a dynamic IoT context. With the growth of clients and data in IoT systems and services, it is troublesome to initialize IFL and keep an up-to-date and modest model by handling both fresh and stale content. By implementing the stage transition and client selection strategies, the periodic learning in IFL can be triggered rationally with a set of QCs preserving both old and new information

TABLE III
VALUES OF MAC AND SAC AT EACH STAGE IN DIFFERENT SCENARIOS FOR COMPLICATED TASKS (INCLUDING CIFAR-10 AND GTSRB)

Task	Scenario	Method	Stage 1		Stage 2		Stage 3		Stage 4		Stage 5	
			MAC↑	SAC↑	MAC↑	SAC↑	MAC↑	SAC↑	MAC↑	SAC↑	MAC↑	SAC↑
CIFAR-10	1	SLMFed	23.82	0.2147	**39.85**	**0.3699**	**44.43**	**0.4227**	**45.93**	**0.4293**	**54.36**	**0.5032**
		FedAvg	24.8	0.2186	36.31	0.3437	37.96	0.3622	44.15	0.4251	48.32	0.4478
		FedProx	24.07	0.2244	28.19	0.2641	28.85	0.2816	29.54	0.2722	31.26	0.2645
		NOMA	24.8	0.2186	36.31	0.3437	37.3	0.3577	44.24	0.4243	46.48	0.4294
		FedProf	21.32	0.1843	25.06	0.2288	30.88	0.2917	41.31	0.3666	43.29	0.42
		FedLAMA	22.69	0.2058	35.96	0.3308	42.57	0.3934	43.25	0.4049	44.34	0.4262
		FedNTD	**27.7**[*]	**0.2342**	39.48	0.3657	37.03	0.329	36.38	0.3518	36.5	0.3443
		SoteriaFL	23.79	0.2115	38.04	0.3623	38.32	0.3656	42.86	0.4011	43.63	0.4116
		Oort	26.25[†]	0.2331	28.43	0.2663	36.74	0.3512	42.37	0.4041	43.03	0.4032
		E3CS	22.88	0.1953	37.32	0.3364	36.45	0.3452	40.11	0.3882	48.7	0.4757
		FedBalancer	24.8	0.2186	37.84	0.3477	43.38	0.3943	43.78	0.4253	52.88	0.4903
		FedSI	17.56	0.1022	26.05	0.2148	28.63	0.2333	32.37	0.3095	33.45	0.3007
		CFedSI	22.41	0.1463	32.55	0.2818	34.9	0.3257	37.07	0.3482	42.14	0.3805
	2	SLMFed	23.82	0.2147	35.64	0.3409	**43.67**	**0.4055**	**49.55**	**0.4668**	**55.78**	**0.523**
		FedAvg	24.8	0.2186	33.59	0.3186	40.73	0.3715	46.41	0.4414	53.75	0.5201
		FedProx	24.07	0.2244	28.44	0.2805	33.03	0.3086	30.01	0.2943	32.02	0.3151
		NOMA	24.8	0.2186	33.59	0.3186	40.4	0.3646	44.37	0.4205	47.13	0.4458
		FedProf	21.32	0.1843	25.19	0.2367	36.76	0.3403	44.13	0.4217	45.67	0.4405
		FedLAMA	22.69	0.2058	31.09	0.2971	38.32	0.3671	43.9	0.4091	43.37	0.4016
		FedNTD	**27.7**	**0.2342**	32.79	0.2999	34.41	0.3107	33.54	0.3173	35.3	0.3328
		SoteriaFL	23.79	0.2115	33.09	0.32	43.17	0.4048	47.56	0.4489	47.17	0.4391
		Oort	26.25	0.2331	27.8	0.2594	36.4	0.3434	45.95	0.439	52.26	0.4853
		E3CS	22.88	0.1953	37.32	0.3364	36.45	0.3452	40.11	0.3882	48.7	0.4765
		FedBalancer	24.8	0.2186	**37.84**	**0.3477**	43.38	0.3943	43.78	0.4253	52.88	0.4901
		FedSI	18.83	0.1507	23.9	0.2099	26.08	0.2292	28.78	0.2661	34.23	0.2937
		CFedSI	19.49	0.1304	27.12	0.2307	30.91	0.2838	37.5	0.3548	39.02	0.3543
GTSRB	1	SLMFed	33.17	0.1946	51.97	0.4545	**59.83**	**0.5686**	**67.3**	**0.6461**	**78.11**	**0.7476**
		FedAvg	31.45	0.195	47.58	0.361	57.58	0.5367	63.31	0.5898	73.19	0.7002
		FedProx	14.69	0.1284	22.21	0.2074	24.29	0.2366	26.27	0.2547	27.78	0.2709
		NOMA	31.45	0.195	47.58	0.361	57.58	0.5367	63.18	0.5855	72.35	0.6474
		FedProf	**42.93**	**0.2679**	46.58	0.4328	40.52	0.3929	41.13	0.4027	47.79	0.4605
		FedLAMA	21.87	0.1378	43.38	0.3314	51.95	0.4834	60.12	0.5697	70.14	0.6494
		FedNTD	31.32	0.1925	43.37	0.3504	53.77	0.4887	61.69	0.5776	76.17	0.6956
		SoteriaFL	31.71	0.1881	46.15	0.3637	49.01	0.4452	55.33	0.5197	73.49	0.7012
		Oort	32.4	0.2058	**54.33**	**0.4766**	51.96	0.4749	55.6	0.4776	66.98	0.5753
		E3CS	22.34	0.1444	38.84	0.3396	53.48	0.458	62.54	0.5887	65.85	0.6299
		FedBalancer	31.45	0.195	42.85	0.4025	59.18	0.5355	62.46	0.5588	72.78	0.6902
		FedSI	17.4	0.1067	24.78	0.2028	30.21	0.2759	36.83	0.2769	36.78	0.2906
		CFedSI	20.29	0.1105	20.43	0.1533	26.74	0.2415	29.79	0.1629	23.05	0.1981
	2	SLMFed	33.17	0.1946	52.3	0.4733	**68.77**	**0.6521**	**73.39**	**0.6544**	**81.45**	**0.7632**
		FedAvg	31.45	0.195	45.95	0.412	57.07	0.5116	66.25	0.6007	74.21	0.6823
		FedProx	14.69	0.1284	21.7	0.1949	22.41	0.1993	23.71	0.2226	27.35	0.2658
		NOMA	31.45	0.195	45.95	0.412	57.07	0.5116	65.86	0.5931	79.99	0.7455
		FedProf	**42.93**	**0.2679**	44.56	0.4004	44.48	0.389	46.06	0.4331	49.14	0.4857
		FedLAMA	21.87	0.1378	43.04	0.3534	49.86	0.4526	60.94	0.5671	64.82	0.5633
		FedNTD	31.32	0.1925	48.66	0.4201	61.44	0.5478	68.49	0.5734	78.17	0.6204
		SoteriaFL	31.71	0.1881	45.2	0.4247	52.99	0.465	59.23	0.4784	64.6	0.5506
		Oort	32.4	0.2058	**55.49**	**0.488**	61.2	0.571	70.98	0.6257	74.08	0.7138
		E3CS	22.34	0.1444	38.84	0.3396	53.48	0.458	62.54	0.5887	65.85	0.6299
		FedBalancer	31.45	0.195	42.85	0.4025	64.18	0.5841	67.46	0.6076	72.78	0.6902
		FedSI	17.4	0.1067	21.71	0.1509	23.83	0.213	35.15	0.3329	41.18	0.2739
		CFedSI	20.29	0.1105	15.14	0.1138	19.12	0.1646	25.63	0.2328	29.81	0.2301

[*] Bolded text indicates the best value among the compared methods in a stage.
[†] Underlined text indicates the second best value among the compared methods in a stage.

that is beneficial for the learning to avoid catastrophic performance dropping, and in turn, deliver a high-user experience.

2) It is efficient to improve IFL learning performance. To build high-performance models, in general, there is a dilemma among model performance, TT, and

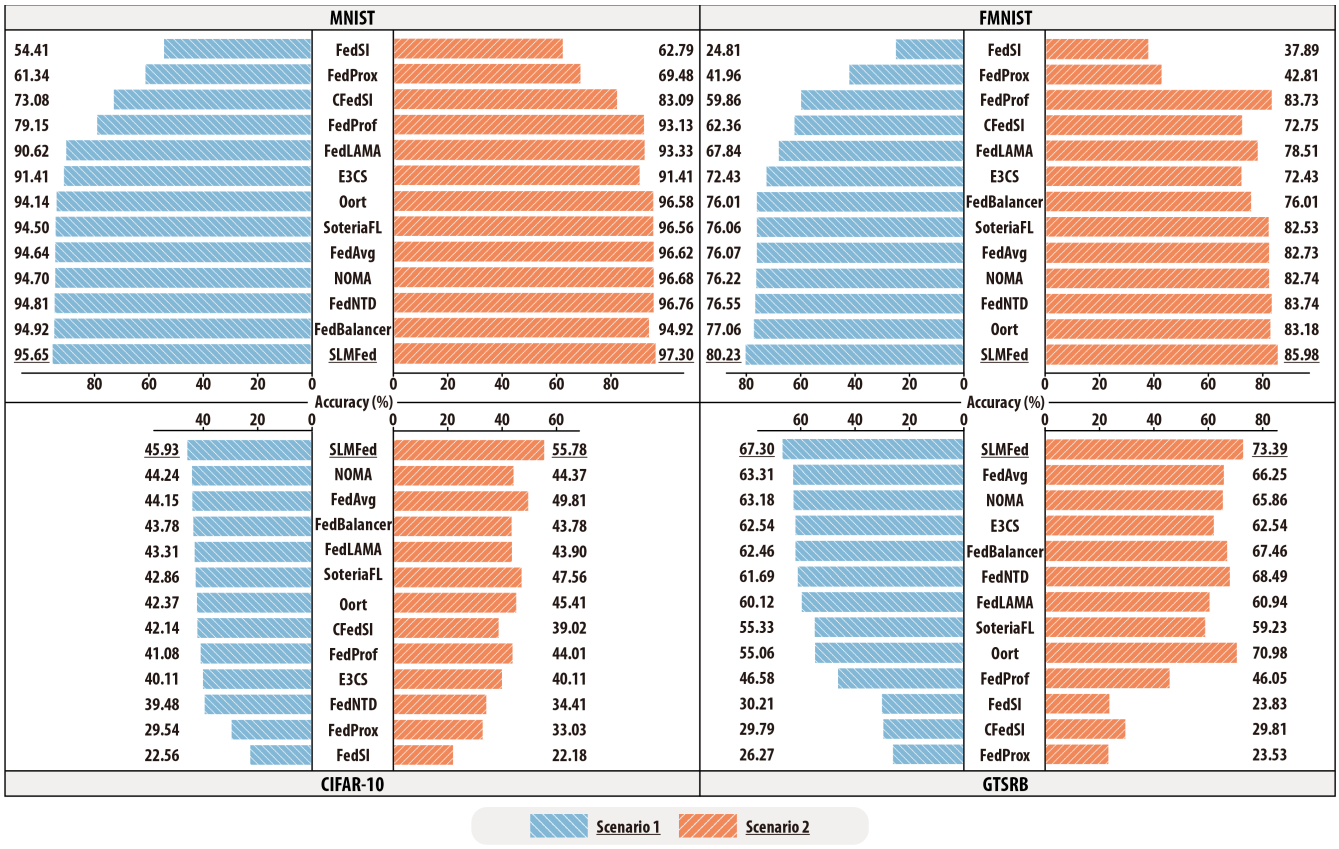


Fig. 7. AC reached under specified CCs, i.e., 6G for MNIST, 17G for FMNIST, 35G for CIFAR-10, and 50G for GTSRB, respectively.

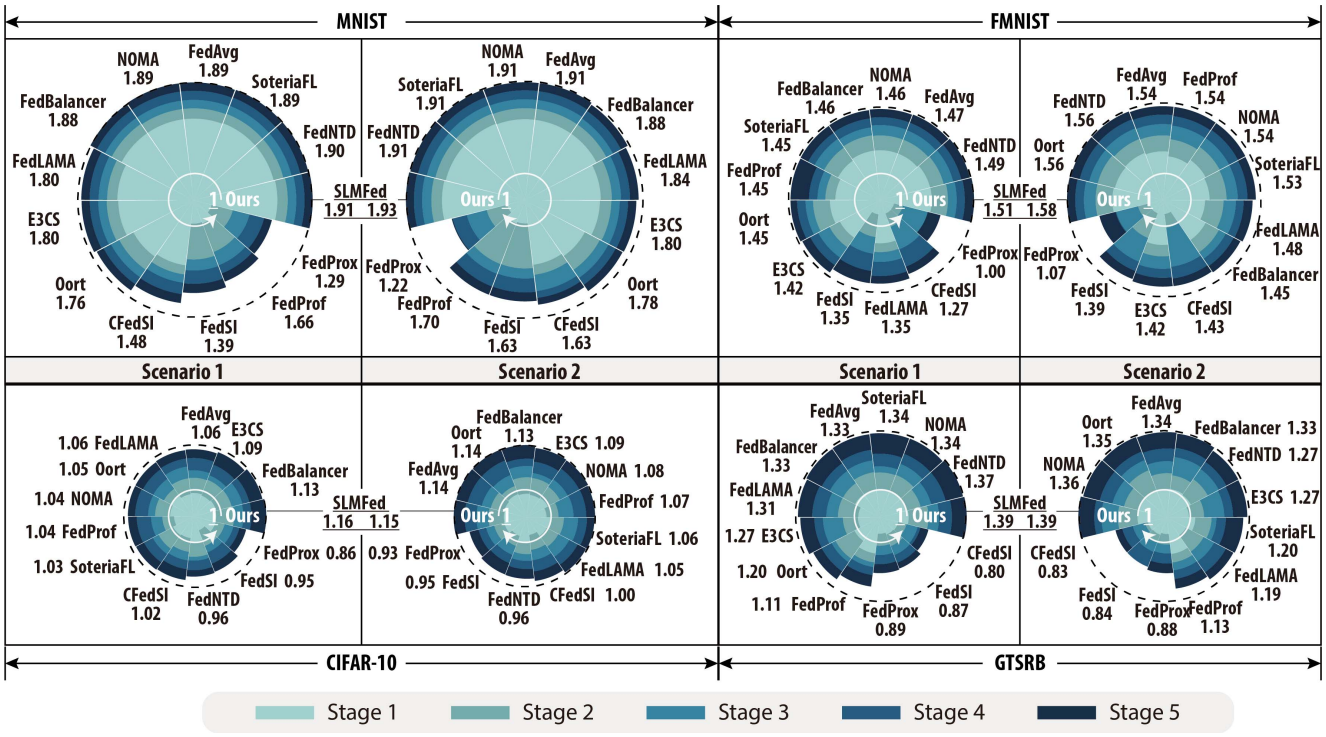


Fig. 8. Nightingale rose chart of SCAC in each task and scenario for all methods. Note that the values are decreasing in the direction of the central white arrow, and the optimal methods are marked with underlines.

learning cost (i.e., in general, higher performance needs more time and cost). Through the layerwise optimization of local model uploading and global model aggregation, which can avoid the overlearning of redundant information and also remedy the impact of non-IID data, the iterative learning in IFL can 1) reduce the

client-server CCs, 2) accelerate the learning process for a model to converge, and 3) improve the overall model performance simultaneously.

3) It is scalable to assist IFL with different complexities. Along with the penetration of IoT systems and services, the learning tasks of IFL will be diversified. Hence, it is essential to assist IFL in these tasks with simplified configurations and procedures, as well as consistent and stable performance. Based on SLMFed, IFL can be implemented and deployed without additional effort, as the decisive parameters used in SLMFed can be auto-configured and self-adjustable to learn high-performance models for tasks with different complexities.

Even though SLMFed can significantly improve the efficiency and effectiveness of IFL, there are still open points to be further explored.

1) *Optimization of Client-Server Collaboration:* The current collaboration mode between the clients and the server may suffer the issue of stragglers, who have relatively limited computational and communicational resources, and in turn, become the performance bottleneck of the whole learning process. Therefore, SLMFed can be enhanced to support an unblocking client-server interaction.
2) *Deployment of Global Model:* The model personalization or localization is not addressed in SLMFed to further elevate the model performance for each client. Hence, by integrating with meta-learning, SLMFed can be applied to learn a global meta-model, based on which, an on-demand update of the local model in each client can be implemented to not only save the learning costs by reducing the number of stages required in IFL but also provide more personalized user experience.
3) *Incentivization of Learning Participants:* The contribution of each client made to the model is not measured and rewarded to maintain an active community for IFL. Accordingly, an incentive mechanism can be investigated and deployed on the client side to acquire and on the server side to assign related rewards rationally and fairly.

V. Conclusion and Future Work

The dynamics of ubiquitous IoT and the ever-growing concerns about data protection drive the integration of IFL to update AI models required in various systems and services along with the growth of clients and data periodically and iteratively. Such that, this article proposes a stage-based and layerwise mechanism for IFL, called SLMFed. Specifically, in periodic learning of IFL, it can rationally determine the stage transition by measuring the quantitative and qualitative impacts of gradually increased clients and accumulated data, and also, heuristically prepare a set of QCs with sufficient old and new data before a new learning stage starts to remedy catastrophic forgetting. Moreover, to elevate the performance of iterative learning in IFL, it can automatically adjust the uploading frequency of model layers in each client according to their RCs before and after the local training, and then, adaptively aggregate received local updates in the server based on their IR to avoid the overleaning issue.

As shown by the evaluation results, SLMFed can achieve high robustness, scalability, and efficiency to support IFL in dynamic IoT contexts. First, it can stabilize the learning to avoid catastrophic performance dropping by using the mechanism of stage transition and client selection, as all the compared methods present converged accuracy curves in all evaluation cases (in total 8). Moreover, it can significantly improve the performance of IFL in terms of model accuracy, CC, and TT. Specifically, compared to the baselines, SLMFed can elevate model accuracy for MNIST, FMNIST, CIFAR-10, and GTSRB by about 8.97%, 17.69%, 35.65%, and 66.05%, respectively, as well as saving CCs by about 150.73%, 142.87%, 34.02%, and 96.15%, and improving stage contribution by about 14.05%, 21.38%, 19.34%, and 33.30%.

In the future, as IoT systems and services, in general, manage vast interconnected things with heterogeneous software and hardware configurations, a flexible mechanism to support an asynchronous client-server collaboration in SLMFed will be studied to remove potential performance bottlenecks caused by the stragglers. Moreover, to cost-efficiently optimize user experience along with the growth of their local data, a model personalization procedure will be investigated to, first, train a general global model, and then, adopt it rapidly to better support diversified users. Finally, it is essential for actual applications of SLMFed to attract more users, who are willing to join the learning and share their knowledge, and hence, incentive mechanisms will be studied to distinguish and reward trustworthy and reputable participants correctly and fairly for the sustainability of IFL.

References

[1] L. You, B. Tunçer, R. Zhu, H. Xing, and C. Yuen, "A synergetic orchestration of objects, data, and services to enable smart cities," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10496–10507, Dec. 2019.
[2] B. Yang et al., "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.
[3] L. Carnevale et al., "Toward improving robotic-assisted gait training: Can big data analysis help us?" *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1419–1426, Apr. 2019.
[4] A. V. Malawade, S.-Y. Yu, B. Hsu, D. Muthirayan, P. P. Khargonekar, and M. A. Al Faruque, "Spatiotemporal scene-graph embedding for autonomous vehicle collision prediction," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9379–9388, Jun. 2022.
[5] W. Saadeh, S. A. Butt, and M. A. B. Altaf, "A patient-specific single sensor IoT-based wearable fall prediction and detection system," *IEEE Trans. neural Syst. Rehabil. Eng.*, vol. 27, no. 5, pp. 995–1003, May 2019.
[6] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.
[7] L. You, J. He, W. Wang, and M. Cai, "Autonomous transportation systems and services enabled by the next-generation network," *IEEE Netw.*, vol. 36, no. 3, pp. 66–72, May/Jun. 2022.
[8] B. Yang et al., "A joint energy and latency framework for transfer learning over 5G industrial edge networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 531–541, Jan. 2022.
[9] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, Apr. 2021.

[10] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10782–10793, Nov. 2020.

[11] H. Xu, L. Zhang, O. Onireti, Y. Fang, W. J. Buchanan, and M. A. Imran, "BeepTrace: Blockchain-enabled privacy-preserving contact tracing for COVID-19 pandemic and beyond," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3915–3929, Mar. 2021.

[12] M. A. Hussain, S.-A. Huang, and T.-H. Tsai, "Learning with sharing: An edge-optimized incremental learning method for deep neural networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 2, pp. 461–473, Apr.–Jun. 2023.

[13] M. F. Criado, F. E. Casado, R. Iglesias, C. V. Regueiro, and S. Barro, "Non-IID data and continual learning processes in federated learning: A long road ahead," *Inf. Fusion*, vol. 88, pp. 263–280, Dec. 2022.

[14] I. Javed et al., "Next generation infectious diseases monitoring gages via incremental federated learning: Current trends and future possibilities," *Comput. Intell. Neurosci.*, vol. 2023, Mar. 2023, Art. no. 1102715.

[15] B. Li, S. Chen, and Z. Peng, "New generation federated learning," *Sensors*, vol. 22, no. 21, p. 8475, 2022.

[16] J. Dong et al., "Federated class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10164–10173.

[17] L. Zhang, G. Gao, and H. Zhang, "Spatial-temporal federated learning for lifelong person re-identification on distributed edges," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Jun. 1, 2023, doi: 10.1109/TCSVT.2023.3281983.

[18] K. Guo, T. Chen, S. Ren, N. Li, M. Hu, and J. Kang, "Federated learning empowered real-time medical data processing method for smart healthcare," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, early access, Jun. 23, 2022, doi: 10.1109/TCBB.2022.3185395.

[19] X. He et al., "Federated continuous learning based on stacked broad learning system assisted by digital twin networks: An incremental learning approach for intrusion detection in UAV networks," *IEEE Internet Things J.*, vol. 10, no. 22, pp. 19825–19838, Nov. 2023.

[20] V. Benzy, A. Vinod, R. Subasree, S. Alladi, and K. Raghavendra, "Motor imagery hand movement direction decoding using brain computer interface to aid stroke recovery and rehabilitation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 12, pp. 3051–3062, Dec. 2020.

[21] Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo, and K. Li, "Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12597–12608, Nov. 2020.

[22] P. Pinyoanuntapong, W. H. Huff, M. Lee, C. Chen, and P. Wang, "Toward scalable and robust AIoT via decentralized federated learning," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 30–35, Mar. 2022.

[23] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends® Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[24] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[25] L. You, S. Liu, Y. Chang, and C. Yuen, "A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24199–24211, Dec. 2022.

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

[27] L. Tu, X. Ouyang, J. Zhou, Y. He, and G. Xing, "FedDL: Federated learning via dynamic layer sharing for human activity recognition," in *Proc. 19th ACM Conf. Embedded Netw. Sens. Syst.*, 2021, pp. 15–28.

[28] 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.

[29] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6357–6368.

[30] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.

[31] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2020, pp. 1698–1707.

[32] Y. Zhang et al., "CSAFL: A clustered semi-asynchronous federated learning framework," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–10.

[33] J. Ding, E. Tramel, A. K. Sahu, S. Wu, S. Avestimehr, and T. Zhang, "Federated learning challenges and opportunities: An outlook," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2022, pp. 8752–8756.

[34] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala, "Asynchronous federated learning for sensor data with concept drift," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2021, pp. 4822–4831.

[35] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–7.

[36] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[37] M. K. Nori, S. Yun, and I.-M. Kim, "Fast federated learning by balancing communication trade-offs," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5168–5182, Aug. 2021.

[38] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12878–12889.

[39] J. Le, X. Lei, N. Mu, H. Zhang, K. Zeng, and X. Liao, "Federated continuous learning with broad network architecture," *IEEE Trans. Cybern.*, vol. 51, no. 8, pp. 3874–3888, Aug. 2021.

[40] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, "EdgeFed: Optimized federated learning based on edge computing," *IEEE Access*, vol. 8, pp. 209191–209198, 2020.

[41] Z. Li, H. Zhao, B. Li, and Y. Chi, "SoteriaFL: A unified framework for private federated learning with communication compression," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 4285–4300.

[42] G. Lee, M. Jeong, Y. Shin, S. Bae, and S.-Y. Yun, "Preservation of the global knowledge by not-true distillation in federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 38461–38474.

[43] S. Lee, T. Zhang, and A. S. Avestimehr, "Layer-wise adaptive model aggregation for scalable federated learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, 2023, pp. 8491–8499.

[44] T. Huang, W. Lin, L. Shen, K. Li, and A. Y. Zomaya, "Stochastic client selection for federated learning with volatile clients," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20055–20070, Oct. 2022.

[45] H. Sun, X. Ma, and R. Q. Hu, "Adaptive federated learning with gradient compression in uplink NOMA," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16325–16329, Dec. 2020.

[46] Y. Wang and B. Kantarci, "Reputation-enabled federated learning model aggregation in mobile platforms," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[47] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. 15th {USENIX} Symp. Oper. Syst. Design Implement. ({OSDI}*, 2021), pp. 19–35.

[48] W. Wu, L. He, W. Lin, and C. Maple, "FedProf: Selective federated learning based on distributional representation profiling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 6, pp. 1942–1953, Jun. 2023.

[49] J. Shin, Y. Li, Y. Liu, and S.-J. Lee, "FedBalancer: Data and pace control for efficient federated learning on heterogeneous clients," in *Proc. 20th Annu. Int. Conf. Mobile Syst., Appl. Services*, 2022, pp. 436–449.

[50] Z. Zhang, Y. Zhang, D. Guo, S. Zhao, and X. Zhu, "Communication-efficient federated continual learning for distributed learning system with non-IID data," *Sci. China Inf. Sci.*, vol. 66, no. 2, 2023, Art. no. 122102.

[51] T. Feng, M. Wang, and H. Yuan, "Overcoming catastrophic forgetting in incremental object detection via elastic response distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9427–9436.

[52] J. Peng et al., "Overcoming long-term catastrophic forgetting through adversarial neural pruning and synaptic consolidation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 4243–4256, Sep. 2022.

[53] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "FedHome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2818–2832, Aug. 2022.

[54] D. Bacciu, D. Di Sarli, P. Faraji, C. Gallicchio, and A. Micheli, "Federated reservoir computing neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–7.

[55] G. Wei and X. Li, "Knowledge lock: Overcoming catastrophic forgetting in federated learning," in *Proc. Pacific–Asia Conf. Knowl. Disc. Data Min.*, 2022, pp. 601–612.

[56] L. You, S. Liu, T. Wang, B. Zuo, Y. Chang, and C. Yuen, "AiFed: An adaptive and integrated mechanism for asynchronous federated data mining," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 14, 2023, doi: 10.1109/TKDE.2023.3332770.

[57] L. You, S. Liu, B. Zuo, C. Yuen, D. Niyato, and H. V. Poor, "Federated and asynchronized learning for autonomous and intelligent things," *IEEE Netw.*, early access, Oct. 9, 2023, doi: 10.1109/MNET.2023.3321519.

Linlin You (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Pavia, Pavia, Italy, in 2015.

He is an Associate Professor with the School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou, China, and also a Research Affiliate with the Intelligent Transportation System Lab, Massachusetts Institute of Technology, Cambridge, MA, USA. He was a Senior Postdoctoral Fellow with the Singapore-MIT Alliance for Research and Technology, Singapore, and a Research Fellow with the Architecture and Sustainable Design Pillar, Singapore University of Technology and Design, Singapore. He published more than 60 journal and conference papers in the research fields of smart cities, service orchestration, multi-source data fusion, machine learning, and federated learning.

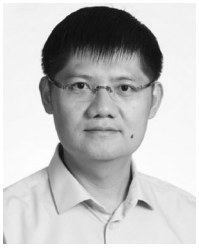
Dr. You is an Associate Editor of *Computer Science* (Springer Nature) and a Young Editor of *The Innovation*.

Zihan Guo received the B.Sc. degree from the School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou, China, in 2023, where he is currently pursuing the master's degree.

His research interests include federated learning and smart cities.

Bingran Zuo received the B.Sc. degree from Zhejiang University, Hangzhou, China, in 1993, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1998.

He is the Deputy Director (Technology) of Rehabilitation Research Institute of Singapore, Nanyang Technological University, Singapore. He was the Program Manager of Future Urban Mobility, Singapore-MIT Alliance for Research and Technology Centre, Singapore. His research interests include robotics and automation, advanced manufacturing technologies, and data-driven robotics with Human-in-the-Loop.



Yi Chang (Senior Member, IEEE) received the Ph.D. degree from the University of Southern California, Los Angeles, CA, USA, in 2016.

He is the Dean of the School of Artificial Intelligence, Jilin University, Changchun, China. He is the author of two books and more than 100 papers in top conferences or journals. He was elected as a Chinese National Distinguished Professor in 2017 and an ACM Distinguished Scientist in 2018. His research interests include information retrieval, data mining, machine learning, natural language processing, and artificial intelligence.

Dr. Chang won the Best Paper Award on KDD'2016 and WSDM'2016. He was the Technical Vice President at Huawei Research America before joining Academia, and the Research Director at Yahoo Labs. He has served as a Conference General Chair for WSDM'2018, SIGIR'2020, and WWW'2025.

Chau Yuen (Fellow, IEEE) received the B.Eng. and Ph.D. degrees from Nanyang Technological University, Singapore, in 2000 and 2004, respectively.

He was a Postdoctoral Fellow with Lucent Technologies Bell Labs, Murray Hill, NJ, USA, in 2005, and a Visiting Assistant Professor with The Hong Kong Polytechnic University, Hong Kong, in 2008. From 2006 to 2010, he was with the Institute for Infocomm Research, Singapore. From 2010 to 2023, he was an Associate Professor with the Engineering Product Development Pillar, Singapore University of Technology and Design, Singapore. Since 2023, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University. He has three U.S. patents and published over 500 research papers at international journals or conferences.

Dr. Yuen was a recipient of the Lee Kuan Yew Gold Medal, the Institution of Electrical Engineers Book Prize, the Institute of Engineering of Singapore Gold Medal, the Merck Sharp and Dohme Gold Medal, and twice a recipient of the Hewlett Packard Prize. He received the IEEE Asia–Pacific Outstanding Young Researcher Award in 2012 and the IEEE VTS Singapore Chapter Outstanding Service Award in 2019. He currently serves as an Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE SYSTEM JOURNAL, and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, where he was awarded as IEEE TNSE Excellent Editor Award and a Top Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY from 2009 to 2015. He also served as the Guest Editor for several special issues, including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE WIRELESS COMMUNICATIONS MAGAZINE, IEEE COMMUNICATIONS MAGAZINE, IEEE VEHICULAR TECHNOLOGY MAGAZINE, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, and *APPLIED ENERGY* (Elsevier). He is a Distinguished Lecturer of IEEE Vehicular Technology Society and also a Highly Cited Researcher by Clarivate Web of Science.