Federated and Asynchronized Learning for Autonomous and Intelligent Things

Linlin You, *Member, IEEE,* Sheng Liu, Bingran Zuo, Chau Yuen*, *Fellow, IEEE,* Dusit Niyato, *Fellow, IEEE,* H. Vincent Poor, *Fellow, IEEE*

***Abstract*—The Internet of Things (IoT) intertwined with autonomous and intelligent things (AITs) is beginning to affect many aspects of our daily lives. Along with this trend, asynchronous federated learning (AFL) is an enabler of harnessing the diverse and heterogeneous sensing and computing capabilities of AITs in a collaborative and privacy-enhancing manner. In this paper, to ease the deployment and improve the performance of AFL for AITs, FedAL (Federated and Asynchronized Learning Framework) is proposed, which can orchestrate the learning process at AITs based on customizable and reusable microservices, activate AITs with high self-information changes as AFL clients to remedy overlearning, optimize the client-server interaction to support cost-efficient model updates, and enhance the model aggregation function by applying an adaptive weight measuring both the information staleness and richness of local updates. It is seen that, compared with three baselines (i.e., FedAvg, FedAsync, and FedConD), FedAL can significantly improve the overall performance in terms of model accuracy by 2.58%, communication delay by 48.83%, and communication cost by 69.84%.**

I. INTRODUCTION

WITH the rapid development of the Internet of Things (IoT), massive autonomous and intelligent things (AITs), e.g., unmanned vehicles, assistive robotics, etc., are being connected to form distributed and versatile networks. By harnessing the plentiful sensing and computing resources of such networks, the level of automation and intelligence of AITs can be further elevated by adopting artificial intelligence (AI) methodology, such as deep neural networks (DNNs) [1]. However, since the centralized learning (CL) approach often requires high-resolution data to be stored and processed at a data center, it is incapable of fully utilizing the distributed computing power and private data of diversified AITs to mine inter-knowledge.

Linlin You is with the School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou 510275, CN, and the Intelligent Transportation Systems Lab, Massachusetts Institute of Technology, MA 02139, USA.

Sheng Liu is with the School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou 510275, CN.

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

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

Dusit Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, SG.

H. Vincent Poor is with the Department of Electrical and Computer Engineering, Princeton University, NJ 08544, USA.

*Corresponding author, e-mail: chau.yuen@ntu.edu.sg

Therefore, a decentralized approach, called Federated Learning (FL), has been proposed to bridge data and computing silos to train shareable models in a collaborative and privacy-enhancing way [2]. Due to its merits in private data protection and training cost reduction, FL has been adopted in several applications, including healthcare, mobility, smart grid, etc. [3]. Moreover, according to the communication mode, FL can be categorized into synchronous FL (SFL) with clients working at the same pace, and asynchronous FL (AFL) with all clients working individually and independently. Since various IoT systems and services are generally distributed at the edge with Non-IID (non-independent and identically distributed) data and heterogeneous computation capabilities and availabilities, AFL is more suitable for AITs to jointly train DNNs with less communication load and latency, as well as fewer learning disruptions and misguidances [4].

Initially, in order to ease bandwidth usage, conventional techniques, i.e., update compression and frequency reduction, have been adopted, and as alternatives to tackle their side effects of information loss, solutions splitting the update of shallow and deep layers of DNNs have also been studied [4], [5]. Even though these methods can reduce communication costs, it is still an unsolved problem to achieve an optimal tradeoff in learning cost and accuracy. Moreover, regarding the variance in temporal staleness and informative richness of local parameters, several weighted strategies have been proposed to update the global model with higher accuracy by steering the learning direction [4]. However, the intrinsic influences of Non-IID data and frequently changed availabilities of AITs have not not addressed thoroughly for AFL from client activation, client-server interaction, then to model aggregation. Finally, since AITs vary from each other in software and hardware capabilities, a scalable and re-deployable AFL framework, such as service-centric networking [6], is missing to unify and simplify the learning configuration.

To implement efficient and effective AFL on diversified and distributed AITs, this article proposes FedAL: Federated and Asynchronized Learning Framework, which has four main contributions:

- It modularizes learning functions of the AFL server and clients as customizable and re-deployable microservices to simplify the deployment process;
- It activates AITs with high self-information changes as AFL clients to train local models with more important parameters for the global model to converge rapidly;
- It uploads deep and shallow layers of DNNs adaptively to achieve an optimal tradeoff in learning cost and accuracy;

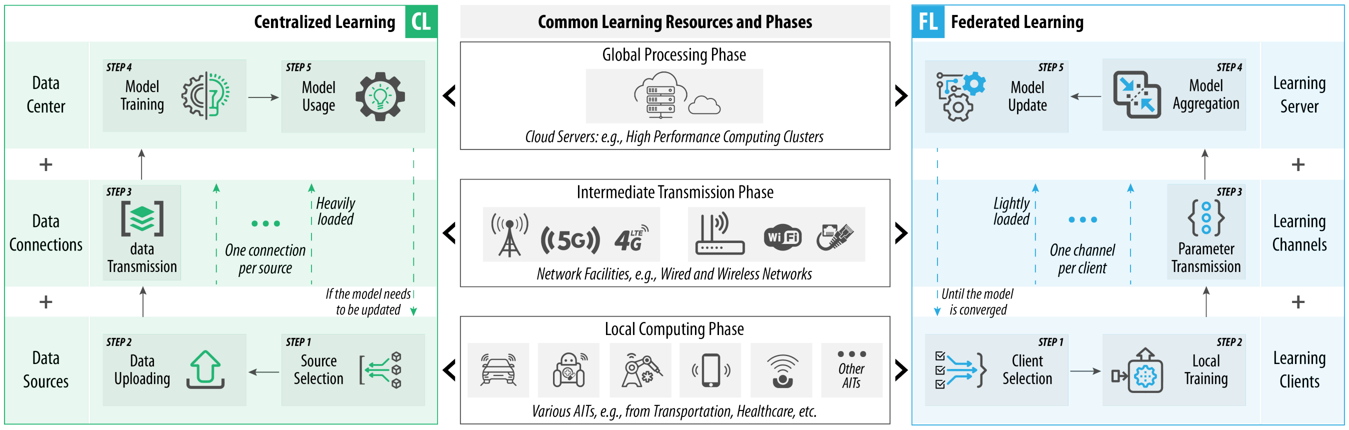

Fig. 1. The comparison of CL and FL. Note that in the middle, there are common learning resources and phases used by both CL and FL

- It enhances the aggregation function based on an adaptive weight measuring both the information staleness and richness of local updates to improve the learning performance.

The remainder of this article is organized as follows. First, Section II compares FL with CL, and then discusses FL in synchronous and asynchronous modes together with emerging challenges and related solutions to disclose the current research gap. After that, FedAL is presented in Section III and evaluated in Section IV, respectively. Finally, Section V concludes the work and sketches future research directions.

II. From Centralized Learning to Federated Learning: Comparisons, Modes, Challenges, and Solutions

The application of advanced technologies and the engagement of laws and regulations about data security and user privacy jointly stimulate a transformation of the learning paradigm from CL to FL to integrate vast and isolated data and also harness plentiful and diverse computing powers of AITs. To illustrate the influence of such a transformation, this section compares the workflows of CL and FL, and then discusses the synchronous and asynchronous modes of FL together with emerging challenges and related solutions.

A. Comparisons between CL and FL

As shown in Figure 1, both CL and FL rely on the same resources from the edge to the cloud, including AITs, networking facilities, and cloud servers, and also consist of three phases, i.e., local computing, intermediate transmission, and global processing. However, their differences can be highlighted through their workflows. As shown on the left side of Figure 1, CL addresses the heterogeneity of the multi-source data for a global model in a data center. First, data sources are selected according to training requirements, and then, related data are uploaded to the data center. After that, the global model is trained at the server and applied at AITs to assist domain-specific tasks, e.g., diagnosis support in healthcare or object identification in mobility applications. Note that according to the feedback gathered during the actual usage, another round of training can be performed to update the model on demand.

In contrast to CL, as shown on the right side of Figure 1, FL implements a decentralized paradigm to train a shareable model in an iterative manner. First, AITs fitting the overall learning objective are activated as FL clients, and then, local models are trained based on their local resources. Instead of explicitly uploading the raw data, local models are uploaded to the server and used to generate a new global model based on a pre-defined aggregation function, e.g., FedAvg [7]. Finally, based on a performance test, the server will determine whether to update the current global model or stop the learning by broadcasting control commands to the clients.

In summary, compared to CL, FL has the following advantages. First, its clients can process their private data locally and communicate with the server cryptographically [8] to ensure user privacy and data security. Second, FL can maintain light network traffic by transmitting only the learning parameters. Third, it can utilize distributed computing powers of AITs to ease the burden of central servers. Finally, it is more suitable to process large-scale and unbalanced Non-IID data isolated at AITs.

B. FL Synchronous and Asynchronous Modes

As shown in Figure 2, FL can foster a learning consortium that covers a wide range of AITs, e.g., unmanned vehicles, assistive robots, robotic arms, mobile devices, and sensing units [9]. FL can also support with various connectivities, e.g., in V2X (vehicle to everything), fiber connections for roadside units, and 6G and Bluetooth for on-board units [10]. To fully utilize these available AITs, FL can work synchronously or asynchronously in two phases, i.e., an initialization phase to activate and organize FL clients, and an execution phase to support the client-server interactions.

1) Initialization Phase: Both synchronous FL (SFL) and asynchronous FL (AFL) need to select and activate available AITs as FL clients according to rules and criteria defined by learning tasks, e.g., to train a traffic sign detector or a lesion annotator, the rules can be selecting top N clients with the highest data quality and computing powers, and the criteria may include the minimum sample size, network bandwidth,

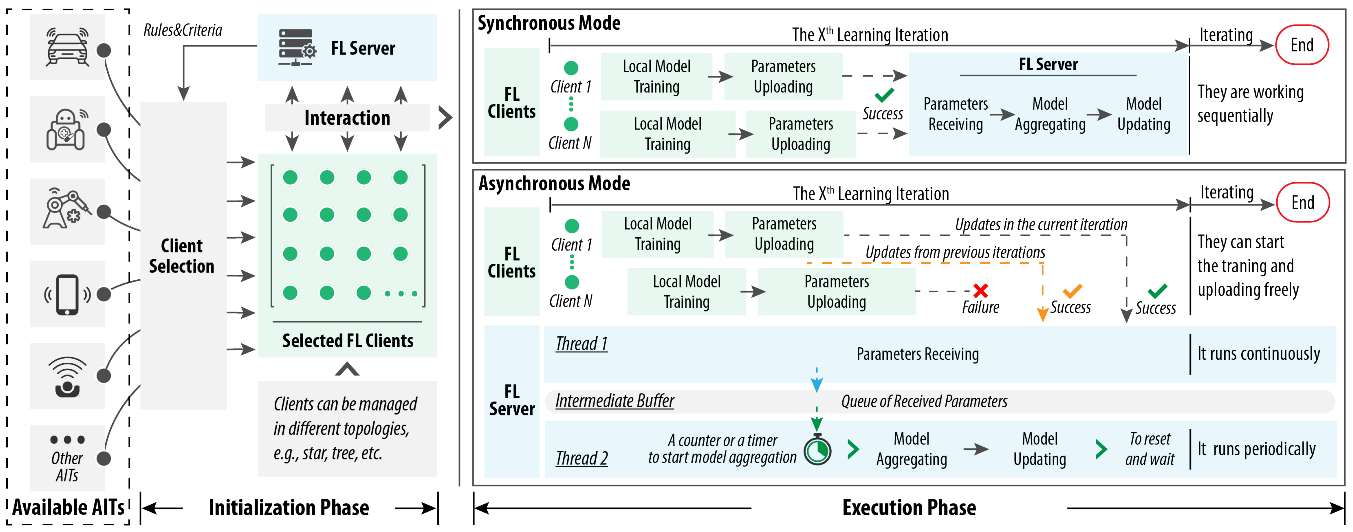

Fig. 2. The similarities and differences between SFL and AFL. They have a common learning consortium of available AITs, and also two working phases, i.e., 1) Initialization phase to activate clients according to predefined rules and criteria, and 2) Execution phase in synchronous or asynchronous modes.

etc. Moreover, since the availability of AITs may change over time and place, it is practical to maintain a sufficient number of learning participants by selecting FL clients periodically. Finally, as a best practice to save communication costs, activated AITs can be organized in various network topologies according to their actual running statuses [9].

2) Execution Phase: The interaction between the server and clients highlights the differences between SFL and AFL.

- *Synchronous mode: All FL clients work at the same pace.* After a start command from the server is received, FL clients will train and then upload local models. Meanwhile, the server will wait for the arrival of all the parameters and aggregate them to update the global model. After that, a new learning iteration starts or the learning ends when a pre-defined target or constraint (e.g., the minimum error rate or maximum iteration) is reached.
- *Asynchronous mode: All FL clients can work separately.* The server can update the global model as soon as it receives a certain number of parameters or triggers a default timer. Hence, two concurrent threads are required, i.e., one thread supports the continuous uploading of local parameters, and the other thread supports the unblocked aggregation of local parameters. It is worth noting that an intermediate buffer exists in between the two threads to store received local parameters.

Compared to AFL, SFL is simpler but limited to supporting diversified AITs, as it requires all participants to be online until the learning ends [2]. Since, in reality, the number of AITs can also grow gradually [4], it can significantly increase the collaboration complexity of SFL to train an efficient and effective model based on scarce and biased data. Even though such complexity can be reduced by the synchronization between FL clients and the server, SFL still suffers from the issue of stragglers, which can lag the whole learning process, leading to unexpected droppings of learning performance [9]. In contrast, AFL has intrinsic advantages in tackling these issues regarding the intermittency of connections, the variability of local information, and the unreliability of learning participants, especially when equipped with dedicated strategies in related learning steps, e.g., in client selection to activate clients with high contributions [11], in parameter transmitting to upload local parameters with scheduled submodels [5], and in model aggregation to process parameters based on fading weights [4].

C. Emerging challenges in AFL

The need for AFL is surging along with the penetration of ubiquitous IoT systems and services, as there are massive AITs with distinguishable data and configurations. As such, several challenges are emerging:

- *C.1 Function orchestration*: Different from conventional solutions, AFL requires related functions to be executed at each client stably and consistently. Therefore, modularized functions are needed to orchestrate a unified and compatible learning process across AITs [6].
- *C.2 Client activation*: If a learning cluster is initialized to use clients randomly, the clients with biased data may cause the over-learning issue. Hence, it becomes critical to activate AITs with sufficient data as AFL clients [11].
- *C.3 Interaction optimization*: The incremental exchanges of learning parameters may disrupt local services and network usage at AITs. Thus, the asynchronous client-server interaction shall be optimized to reduce communication costs, and in turn, improve service quality [4].
- *C.4 Aggregation enhancement*: Besides the informative difference derived from Non-IID data, local parameters from AITs may also vary in staleness. Consequently, how to aggregate them becomes essential for AFL to train high-performance global models [13].

D. Related solutions about AFL

To address these challenges, several solutions are proposed. First, service-centric architectures are discussed as a preferable choice to deploy AI on AITs [6]. Even though related solutions, such as service-oriented architecture, have been utilized,

TABLE I
THE OVERALL EVALUATION OF REVIEWED SOLUTIONS
(● SUPPORTED ○ NOT SUPPORTED)

Solutions	Function Orchestration	Client Activation	Interaction Optimization	Aggregation Enhancement	Highlights (+ pros and - cons)
[6]	●	●	○	○	A software-defined service-centric networking framework: + Service-centric to optimize the learning workload - Weak support for interaction and aggregation
[11]	○	●	○	○	A quality-aware client-selected federated learning algorithm: + Evaluating learning quality to select clients - Behindhand to newly sensed data
[12]	○	○	●	○	Asynchronous transmission scheduling algorithms: + Adaptive transmission scheduling for AFL - Over-learning issue caused by Non-IID not addressed
[5]	○	○	●	●	A resource-efficient learning approach: + Training and uploading the assigned submodel per client + Layerwise gradient information aggregation - Layer uploading frequency not optimized - Over-learning issue undiscussed
[13]	○	○	○	●	An FL approach with resorting to mutual information + Local training and global aggregated steered - Over-learning issue caused by stragglers not tackled
FedAL (Proposed)	●	●	●	●	A unified and compatible AFL framework: + Microservice orchestration for AFL server and clients + Self-information change-based client activation + Adaptive layer uploading + Fused aggregation weight

their actual usage in AFL is rarely observable, showing a clear gap in orchestrating learning functions at diversified AITs. Second, strategies measuring the diversity of local resources are proposed to select an appropriate set of learning participants to tackle over-learning issues, e.g., AUCTION (Automated and qUality aware Client selecTION framework) [11] activating clients with high-quality data within a limited budget. However, in general, these solutions rely heavily on performance tests of trained models, which may make them behindhand to newly sensed data. Third, to reduce the network load, three approaches are discussed, i.e., update compression, frequency reduction, and model split [5]. Compared with the first two approaches, the last approach can avoid side effects of information loss, e.g., a layer-wise asynchronous model update strategy with a reduced update frequency of deep layers of DNNs [5]. However, how to achieve optimal performance in both learning cost and accuracy is still open for solutions. Finally, as for the model aggregation, since the information richness and freshness of local parameters may vary among AFL clients, several weighing strategies are proposed to harness them for a performance boost, e.g., heuristic weights to control the training speed [4], and a mutual information-driven mechanism to steer the training direction [13]. However, how to measure the informative and temporal attributes jointly is still missing.

As summarized in Table I, a framework that can comprehensively address the emerging challenges is required. Therefore, this paper presents FedAL, which can orchestrate microservices at AITs to train DNNs efficiently and effectively.

III. PROPOSED FEDAL

As shown in Figure 3, FedAL implements a unified AFL process consisting of four steps.

A. Microservice Deployment

FedAL contains a microservice registry to manage reusable and re-deployable function units, e.g., 1) self-activation, local training, and parameter uploading microservices for clients, and 2) parameter receiving and aggregation microservices for the server. Moreover, according to the task specification, related microservices can be extracted, customized, and deployed to the clients and server beforehand. Hence, a collaborative AFL process can be orchestrated without excessive efforts on reengineering and redevelopment.

B. Client Activation

Through daily usage, AITs can continuously sense new data and wait for activation. To select clients from available AITs, an indicator measuring the self-information changes of AITs is used. Specifically, the information changes can be computed based on relative entropy or Kullback–Leibler Divergence of sensed datasets of AITs at two different times, e.g., in the $t-1$ and t learning interactions. Accordingly, clients with changes in the top α percent will be selected to start the local training, and for those unselected, they can continue sensing new data and waiting for the activation in future iterations. Note that α is a hyperparameter, which can be configured according to the grid search method or a heuristic optimizer that can adjust α according to the complexity of tasks and the richness of data.

C. Client-Server Interaction

Based on protected communication channels (e.g., encrypted or blockchain-based [8]), the client-server interaction in FedAL is unbounded. Hence, local parameters can be on-time received, late received, or lost. If local parameters are with the first two statuses, they are well received by the server,

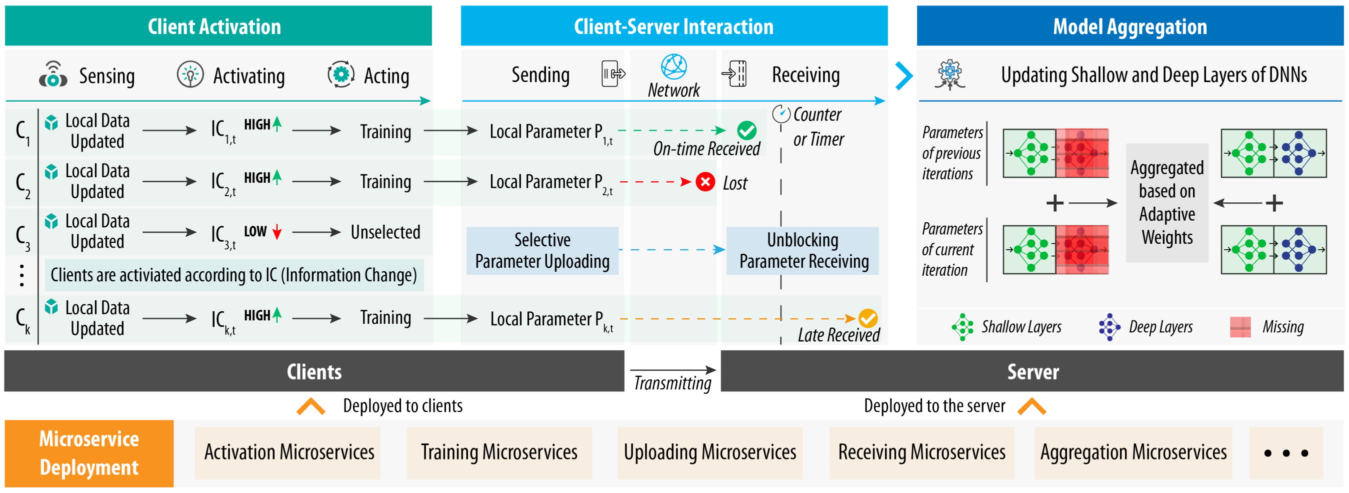

Fig. 3. The overall architecture and workflow of FedAL. It includes (A) Microservice Deployment to load learning-related functions on AFL clients and the server; (B) Client Activation to select clients with high information changes; (C) Client-Server Interaction to upload the deep and shallow layers selectively; and (D) Model Aggregation to merge received local parameters based on adaptive weights.

but with a difference in the arrival time, either before or after a default counter or timer is triggered. Moreover, based on the observation that shallow layers of DNNs (i.e., convolutional layers) are more crucial but with fewer parameters than deep layers (i.e., fully connected layers) [4], an adaptive parameter uploading strategy, noted as $APU(m, \beta)$, is designed to reduce the communication cost by uploading deep layers adaptively. Specifically, APU segments the whole learning process into several learning phases, in each of which, there are m learning iterations, and deep layers are uploaded in the last $\lceil \beta \times m \rceil$ iterations. Note that m and β are the hyperparameters, which can be set either manually according to prior knowledge or dynamically according to the runtime feedback generated from a critic (that can be pre-trained based on meta-learning).

D. Model Aggregation

Since received parameters can have different levels of staleness and amount of information, an adaptive aggregation mechanism is implemented to update the global model by using an adaptive weight (AW). As for the calculation of AW_i for the i_{th} client, first, a temporal weight TW_i is computed based on the differences between the created time and received time of the local model. Second, an informative weight IW_i is calculated by the information entropy of local data used to train the local model. Finally, TW_i and IW_i are multiplied and normalized among all clients to generate AW_i.

In summary, compared to the current solutions, first, FedAL can ease the configuration of the learning process by deploying microservices onto its clients and server. Second, it can avoid overlearning issues by selecting AITs with appropriate data as clients. Third, it can reduce communication costs without damaging model performance by optimizing client-server interactions. Finally, it can enhance the model aggregation by jointly weighing the temporal and informative attributes.

IV. Case Study and Discussion

FedAL is evaluated together with three baselines, i.e., FedAvg [7] a widely used synchronous method, FedAsync [14] a classic asynchronous algorithm, and FedConD [15] an AFL method considering concept drift caused by continuously increased sensing data. Note that FedAL has been integrated and publicly available through an open-source project [1].

A. Simulation Setup

First, 50 AITs are visualized with various capabilities (reflected by the data transmission time ranging from 10s to 40s with a dropping rate ranging from 1% to 5%) to train a convolutional neural network (CNN) with 2 convolutional layers (shallow layers) and 2 fully connected layers (deep layers). Specifically, the two convolutional layers have 32 and 64 channels, respectively, followed by a 2×2 max-pooling layer. The fully connected layer has 256 units followed by a softmax unit as the output layer. The default local epoch, learning rate, and batch size are 2, 0.003, and 48, respectively.

Second, gradually increasing and Non-IID data per client is created based on the training samples of five standard datasets, i.e., MNIST[2], FMNIST[3], German Traffic Sign Benchmark (GTSRB)[4], CIFAR-10[5], and Driver Distraction Detection (3D) dataset[6]. MNIST and FMNIST are two widely used classification datasets about handwritten digits and fashion images, respectively. As for GTSRB, it consists of 43 classes, e.g., speed limit, children crossing and ahead only. CIFAR-10 and 3D datasets contain 10 classes about traffic means/animals (such as airplanes and birds) and driver statuses (such as safe driving and talking to passengers), respectively. In general, the local data of AITs will have an initial portion of about 6% to 12%, and increase by 2%-4% or 0.2%-0.4% per learning iteration to simulate the arrival of new data in the real world. Note that the test samples of the five datasets are kept untouched and only used to evaluate model accuracy.

[1]https://github.com/IntelligentSystemsLab/generic_and_open_learning_federator
[2]http://yann.lecun.com/exdb/mnist
[3]https://github.com/zalandoresearch/fashion-mnist
[4]https://bitbucket.org/jadslim/german-traffic-signs
[5]https://www.cs.toronto.edu/ kriz/cifar.html
[6]https://www.kaggle.com/c/state-farm-distracted-driver-detection

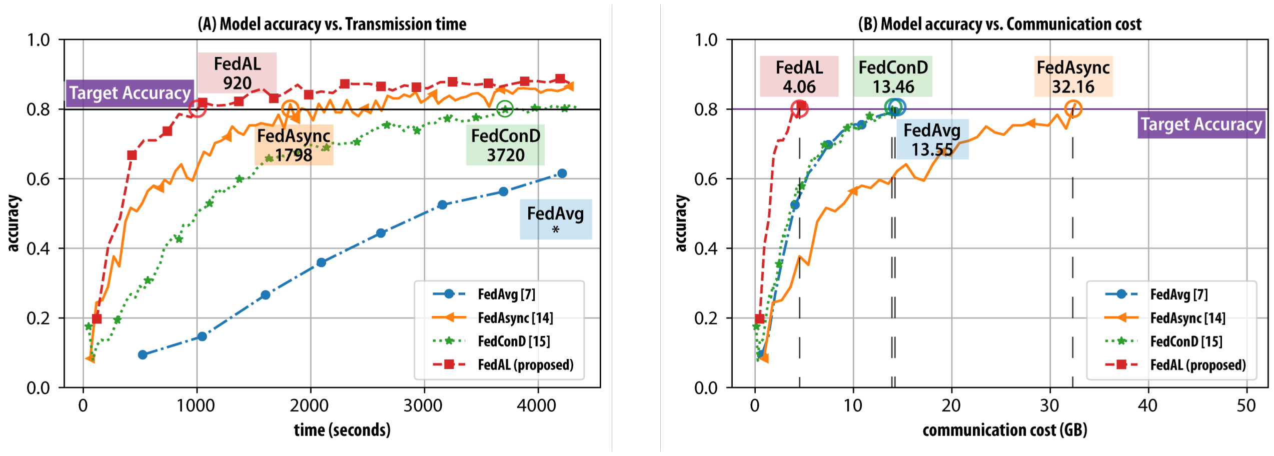

Fig. 4. Comparison between FedAL and three baselines on GBSRB dataset. (A) Model accuracy vs. Transmission time; and (B) Model accuracy vs. Communication cost. It shows that the proposed FedAL can achieve the target accuracy more rapidly with fewer communication costs.

Finally, FedAvg (with C in [7] = 0.1), FedAsync (with a and μ in [14] = 0.5 and 1, respectively), FedConD (with γ in [15] = 0.2), and the proposed FedAL are compared through three evaluation metrics, namely model accuracy, communication cost, and transmission time. Specifically, the model accuracy is tested on the global model updated at the server by using the test samples. The communication cost accumulates the size of local parameters transmitted over the network. The transmission time is the sum of the waiting time to start an aggregation in the server, which also reflects the overall communication delay.

B. Results and Discussion

FedAL is evaluated with the three baselines to reveal related improvements achieved in transmission time, communication cost, model accuracy, and learning stability.

1) Transmission Time: As shown in Figure 4 (A), a minor delay is experienced in AFL methods, as the stragglers, which are the bottlenecks of FedAvg in SFL, can be addressed by the unblocking client-server interaction. While comparing FedAsync and FedConD with FedAL, a clear gap in accuracy growth rate and stability can be observed. Moreover, when the target accuracy of 80% is first achieved, the delay of FedAL (920s) is approximately 48.83% lower than that of the second-best method FedAsync (1798s).

2) Communication Cost: As shown in Figure 4 (B), FedAsync consumes more communication resources to be comparable with the rest three methods. Another AFL method FedConD shares a similar learning curve with the SFL method FedAvg, which indicates the superiority of selective communication strategy. While comparing the communication cost of FedAL (4.06GB) at the target accuracy with the second-best method FedConD (13.46GB), a reduction of 69.84% can be further achieved since the local model parameters are selectively uploaded and adaptively aggregated in FedAL.

3) Model Accuracy: As shown in Figure 4 (A), when the maximum learning time is exceeded, FedAL can reach the highest model accuracy of 88.74%, with an improvement of about 44.20%, 2.58%, and 9.54% compared to FedAvg (61.54%), FedAsync (86.51%), and FedConD (81.01%), respectively.

4) Learning Stability: As illustrated in Table II, the local accuracies of FedAL in various settings of the three hyper-parameters, i.e., α, m, and β, are compared with the three baselines. First, as for $APU(m, \beta)$, it is correlated to the learning tasks and more complicated tasks may need a higher m and lower β to achieve the best performance. Second, FedAL with $\alpha = 0.7$ outperforms the baselines and other FedAL settings on three datasets (except for CIFAR-10 with $\alpha = 0.8$). The results indicate that an appropriate number of activated clients controlled by α is essential for FedAL to achieve optimal performance.

In summary, due to its ability in selecting clients with sufficient data and self-information changes, transmitting local updates with a distinguishable and adjustable frequency, and updating the global model according to adaptive weights that measure the staleness and richness of received local parameters in each asynchronous learning round, FedAL outperforms the three baselines with higher learning accuracy and stability, as well as lower communication delay and cost.

V. Conclusions

This article has first compared FL with CL to elucidate its advantages in accommodating distributed AITs in a collaborative and privacy-preserving way. Moreover, AFL has been compared with the widely discussed SFL to reveal its merits in supporting diversified AITs with heterogeneous configurations. After that, by identifying encountered challenges and flaws of related solutions about AFL to support AITs, FedAL has been proposed, which implements a unified AFL framework with microservice deployment, client activation, client-server interaction, and model aggregation to train DNNs efficiently and effectively. Finally, a case study simulating a real-world scenario of AITs has been given to comparing FedAL with three baselines, i.e., FedAvg, FedAsync, and FedConD. It shows that compared to the second best method, FedAL can improve the performance significantly, i.e., increasing model accuracy by 2.58%, reducing communication delay by 48.83%, and saving communication cost by 69.84%.

TABLE II
LOCAL ACCURACIES OF COMPARED METHODS

Method		Dataset*				
		MNIST	FMNIST	GBSRB	CIFAR-10$^{\$}$	3D
FedAvg		94.41±0.04%	61.42±0.11%	74.39±0.10%	<u>49.06±0.03%</u>	83.38±0.45%
FedAsync		95.72±0.03%	67.62±0.24%	78.38±0.77%	34.57±0.26%	86.16±1.51%
FedConD		<u>96.91±0.10%</u>†	<u>71.87±0.21%</u>	<u>83.70±0.10%</u>	48.51±0.05%	<u>91.82±0.63%</u>
FedAL	α**=0.6**	98.16±0.41%	72.04±0.11%	87.52±0.09%	49.39±0.18%	93.83±0.44%
	α**=0.7**	**98.73±0.21%**§	**72.69±0.08%**	**89.72±0.03%**	49.87±0.15%	**94.21±0.30%**
	α**=0.8**	98.50±0.14%	72.23±0.11%	87.70±0.07%	**50.68±0.14%**	92.05±0.17%

*For MNIST, FMNIST, GBSRB, CIFAR-10, and 3D datasets, the best settings of $APU(m,\beta)$ in FedAL are $APU(3,\frac{1}{3})$, $APU(4,\frac{1}{4})$, $APU(6,\frac{1}{6})$, $APU(3,\frac{1}{3})$, and $APU(4,\frac{1}{4})$, respectively, according to the preliminary experimental results. In addition, a counter is used in FedAL to control the asynchronous model aggregation, which makes the number of actual participants in each round the same as FedAvg for a fair comparison.
$^{\$}$The results of CIFAR-10 are not competitive, as CIFAR-10 presents a more complex classification task compared to the rest four datasets, and the common CNN model used in the evaluation is relatively simple with two convolutional layers and two fully connected layers.
†Numbers with underlines are the best values achieved by the baselines.
§Bold numbers are the best performance among all methods.

This study suggests several interesting research directions that can be:

- Easy-to-install containers: The running of microservices at AITs depends on a visualized container, which is still burdensome to install. Therefore, a simplified procedure can be designed and implemented to further lift the usability and compatibility of FedAL;
- Global data quality indicators: Even though the adaptive client activation and model aggregation implemented in FedAL can significantly remedy the over-learning issue, the impact of Non-IID data still remains as currently the quality of data is measured in isolation. Hence, indicators with global probes in AITs can be investigated to fully resolve the over-learning issue in FedAL;
- Cost-effective incentive mechanisms: It is unfair for participants who contribute the most to be treated the same as slackers. Moreover, it is ideal for collaborators to maximize benefits while minimizing costs. Thus, incentive mechanisms can be studied to foster an active and reputable consortium of AITs for FedAL.

ACKNOWLEDGMENT

The research is supported by the National Natural Science Foundation of China (62002398) and the GuangDong Basic and Applied Basic Research Foundation (2023A1515012895).

REFERENCES

[1] I. A. Ridhawi, S. Otoum, M. Aloqaily, and A. Boukerche, "Generalizing ai: Challenges and opportunities for plug and play ai solutions," *IEEE Network*, vol. 35, no. 1, pp. 372–379, 2021.
[2] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Wireless communications for collaborative federated learning," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 48–54, 2020.
[3] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.
[4] 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 of Things Journal*, vol. 9, no. 23, pp. 24 199–24 211, 2022.
[5] R. Yu and P. Li, "Toward resource-efficient federated learning in mobile edge computing," *IEEE Network*, vol. 35, no. 1, pp. 148–155, 2021.
[6] X. Li, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Advancing software-defined service-centric networking toward in-network intelligence," *IEEE Network*, vol. 35, no. 5, pp. 210–218, 2021.
[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.
[8] N. R. Pradhan, A. P. Singh, S. Verma, M. Wozniak, J. Shafi, and M. F. Ijaz, "A blockchain based lightweight peer-to-peer energy trading framework for secured high throughput micro-transactions," *Scientific Reports*, vol. 12, no. 1, p. 14523, 2022.
[9] H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, and H. V. Poor, "Lead federated neuromorphic learning for wireless edge artificial intelligence," *Nature Communications*, vol. 13, no. 1, pp. 1–12, 2022.
[10] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6g wireless system: Design challenges and solutions," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 40–47, 2021.
[11] Y. Deng, F. Lyu, J. Ren, H. Wu, Y. Zhou, Y. Zhang, and X. Shen, "Auction: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1996–2009, 2022.
[12] H.-S. Lee and J.-W. Lee, "Adaptive transmission scheduling in wireless networks for asynchronous federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3673–3687, 2021.
[13] M. P. Uddin, Y. Xiang, X. Lu, J. Yearwood, and L. Gao, "Mutual information driven federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1526–1538, 2021.
[14] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," in *Proceedings of the 12th Annual Workshop on Optimization for Machine Learning*. opt-ml.org, 2020.
[15] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala, "Asynchronous federated learning for sensor data with concept drift," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 4822–4831.

Linlin You received his Ph.D. from University of Pavia, Italy, in 2016. He is currently an Associate Professor at Sun Yat-sen University and a Research Affiliate at Massachusetts Institute of Technology.

Sheng Liu received his B.E. from Sun Yat-sen University, China, in 2021. He is currently pursuing his master's degree at Sun Yat-sen University.

Bingran Zuo received his Ph.D. degree from Shanghai Jiao Tong University, China, in 1998. He is currently a Principal Research Fellow, Deputy Director at Rehabilitation Research Institute of Singapore, Nanyang Technological University.

Chau Yuen [F] received his Ph.D. degree from Nanyang Technological University, Singapore, in 2004. He is currently an Associate Professor at Singapore University of Technology and Design.

Dusit Niyato [F] received his Ph.D. degree from the University of Manitoba, Canada, in 2008. He is currently a Professor at Nanyang Technological University.

H. Vincent Poor [F] received his Ph.D. degree from Princeton University in 1977. He is currently the Michael Henry Strater University Professor at Princeton.