FedASA: A Personalized Federated Learning With Adaptive Model Aggregation for Heterogeneous Mobile Edge Computing

Dongshang Deng, Xuangou Wu, *Member, IEEE*, Tao Zhang, *Member, IEEE*, Xiangyun Tang, Hongyang Du, Jiawen Kang , *Senior Member, IEEE*, Jiqiang Liu, *Senior Member, IEEE*, and Dusit Niyato, *Fellow, IEEE*

***Abstract*—Federated learning (FL) opens a new promising paradigm for the Industrial Internet of Things (IoT) since it can collaboratively train machine learning models without sharing private data. However, deploying FL frameworks in real IoT scenarios faces three critical challenges, i.e., statistical heterogeneity, resource constraint, and fairness. To address these challenges, we design a fair and efficient FL method, termed FedASA, which can address the challenge of statistical heterogeneity in resource-constrained scenarios by determining the shared architecture adaptively. In FedASA, we first present a cell-wised shared architecture selection strategy, which can adaptively construct the shared architecture for each device. We then design a cell-based aggregation algorithm for aggregating heterogeneous shared architectures. In addition, we provide a theoretical analysis of the federated error bound, which provides a theoretical guarantee for the fairness. At the same time, we prove the convergence of FedASA at the first-order stationary point. We evaluate the performance of FedASA through extensive simulation and experiments. Experimental results in cross-location scenarios show that FedASA outperformed the state-of-the-art approaches, improving accuracy by up to 13.27% with better fairness and faster convergence and communication requirement has been reduced by 81.49%.**

***Index Terms*—Internet of Things, mobile edge computing, personalized federated learning, resource constraint, statistical heterogeneity.**

Manuscript received 16 October 2023; revised 8 August 2024; accepted 15 August 2024. Date of publication 20 August 2024; date of current version 5 November 2024. This work was supported in part by the National Nature Science Foundation of China under Grant 62172003, Grant 92167204, Grant 62072030, and Grant 62302539, in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2022-051, in part by the Open Fund of Anhui Engineering Research Center for Intelligent Applications and Security of Industrial Internet under Grant IASII24-01, in part by the China Postdoctoral Science Foundation under Grant 2024T170047, Grant GZC20230223, and Grant 2024M750165, in part by the National Research Foundation, Singapore, in part by the Infocomm Media Development Authority under its Future Communications Research & Development Programme, Defence Science Organisation (DSO) National Laboratories under the AI Singapore Programme under Grant FCP-NTU RG-2022-010 and Grant FCP-ASTAR-TG-2022-003, in part by Singapore Ministry of Education (MOE) Tier 1 under Grant RG87/22, and in part by the NTU Centre for Computational Technologies in Finance (NTU-CCTF). Recommended for acceptance by N. Zhang. *(Corresponding authors: Xuangou Wu; Tao Zhang.)*

Dongshang Deng and Xuangou Wu are with the School of Computer Science and Technology, Anhui University of Technology, Ma'anshan 243002, China, and also with Anhui Province Key Laboratory of Digital Twin Technology in Metallurgical Industry, Ma'anshan 243002, China (e-mail: dsdeng@ahut.edu.cn; wuxgou@ahut.edu.cn).

Tao Zhang is with the School of Cyberspace Science and Technology, Beijing Jiaotong University, Beijing 100044, China, and also with Anhui Engineering Research Center for Intelligent Applications and Security of Industrial Internet, Beijing 100044, China (e-mail: taozh@bjtu.edu).

Xiangyun Tang is with the School of Information Engineering, Minzu University of China, Beijing 100081, China (e-mail: xiangyunt@muc.edu.cn).

Hongyang Du is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong (e-mail: duhy@eee.hku.hk).

Jiawen Kang is with the School of Automation, Guangdong University of Technology, Guangzhou 510006, China (e-mail: kavinkang@gdut.edu.cn).

Jiqiang Liu is with the School of Cyberspace Science and Technology, Beijing Jiaotong University, Beijing 100044, China (e-mail: jqliu@bjtu.edu.cn).

Dusit Niyato is with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: dniyato@ntu.edu.sg).

I. Introduction

With the advancement of mobile computing technology and the Internet of Things (IoT), industrial digitization and intelligence are rapidly progressing [1], [2], [3]. Intelligent applications in mobile edge computing (MEC) systems are supported by widely deployed edge devices (e.g., sensors) [4], [5]. As the number of sensors deployed increases, the data collected grows exponentially, resulting in a larger accumulation of data at the network edge (e.g., gateways and switches) [6], [7]. However, the network bandwidth burden is an obstacle to uploading data to the cloud for centralized learning [8], [9], [10]. Federated learning (FL) [11] emerges as a solution, allowing edge devices to train machine learning models collaboratively without transmitting private data to the parameter server. As such, FL has many promising applications in IoT, such as human activity recognition [12], [13], fault diagnosis [14], [15], [16], [17], and augmented reality [18].

In MEC, the conventional FL framework consists of two main steps, including training on the local device and model parameter aggregation on the parameter server. Specifically, each device (i.e., the FL worker) uses the gradient descent algorithm to update the model locally and then sends the updated model parameters to the parameter server. The parameter server aggregates the received model parameters by FedAvg [11] and sends the aggregated model to each device for a new round of training. After multiple rounds of training, a trained global model is obtained. This framework demonstrates exemplary performance in ideal settings that include sufficient computing and communication resources on the local device, independent and identically distributed data across devices.

However, in practical IoT scenarios, the following challenges should be considered to achieve an effective and efficient FL framework. **Statistical heterogeneity.** In IoT scenarios, statistical heterogeneity can be categorized into *cross-domain* and *cross-location*. In [14], the data collected from a specific environment is referred to as a device domain, and all devices together constitute cross-domain. Cross-domain heterogeneity is caused by the monitored devices (e.g., production machines) [14], [15], where devices with varying demands operate at different environments. Similarly, cross-location heterogeneity is due to the location of monitoring devices (e.g., sensors) [19], such as the differences in the locating of sensors on devices. **Resource constraint.** Different from datacenters with ample communication resources, in IoT, the communication resources of edge devices are limited [20]. The training process of FL usually requires frequent pushing and pulling of model parameters. Furthermore, as model architectures scale up (e.g., transformers [21]), transmitting the entire model architecture between parameter server and local devices leads to heavy communication requirements (e.g., network bandwidth). **Fairness.** Due to the statistical heterogeneity, there are significant accuracy differences among edge devices [22]. In [23], fairness is used to measure these differences, and a lower standard deviation of accuracy indicates higher fairness. To achieve an effective FL framework, it is necessary to ensure fairness across devices. Note that improving average accuracy while also enhancing fairness is a challenging task, as it requires boosting the performance of all edge devices.

Existing works in the literature that address statistical heterogeneity in FL have fallen into the data-based and the model-based categories. Data-based methods focus on constructing independent and identically distributed datasets to reduce the heterogeneity. In contrast, model-based methods aim to adapt statistical heterogeneity from model architecture or parameter optimization. For data-based methods, Zhao et al. [24] proposes to share a small portion of the global data with each device. FedGen [25] learns a global generator to generate pseudo data and then sends it to each device to manage local updates. FedFTG [26] learns a generator that generates pseudo data with the same distribution as that of each device to optimize the update of the global model. However, transferring data between parameter server and local devices causes additional communication overhead and privacy risks. There are two types of model-based methods based on their purpose: single-model methods and multi-model methods. Single-model methods focus on training a better global model, while multi-model methods look at producing customized models for each local device. Single-model methods care more about the convergence and performance of the global model, e.g., FedProx [27], SCAFFOLD [28], which align the local model with the global model. Conversely, multi-model methods pay more attention to the performance of the model in each device and try to train personalized models. FedFomo [29] computes the first-order approximations of the optimal model weighted combination for each device. Ditto [23] proposes to add a global regularization factor when training personalized models. However, the above methods do not consider the communication resource limitation of the local device.

TABLE I
THE COMPARISON OF THE PREVIOUS WORKS AND OURS

Solutions	Statistical heterogeneity	Resource constraint	Fairness
FedAvg [11]	✗	✗	✗
FedProx [27]	✓	✗	✗
FedGen [25]	✓	✗	✗
FedFTG [26]	✓	✗	✗
Ditto [23]	✓	✗	✓
FedFomo [29]	✓	✗	✓
FedPer [30]	✓	✓	✗
Ours	✓	✓	✓

FedPer [30] reduces communication overhead by uploading only the base layer, while achieving model personalization by retaining a local personalized layer. Table I presents a comparison of the previous works and ours. In short, none of the aforementioned works have addressed these challenges in real IoT scenarios.

In this paper, we aim to address two problems: (i) What are the main statistical heterogeneity factors among edge devices in IoT scenarios? (ii) How do we address statistical heterogeneity while considering resource constraints and fairness in IoT scenarios? For problem (i), our empirical results reveal that cross-location issues can lead to more severe statistical heterogeneity than the widely studied cross-domain issues. The existing methods that can alleviate statistical heterogeneity have different performance degradation when facing cross-location issues in IoT scenarios. These phenomena motivate us to explore new solutions. For problem (ii), to address the cross-location issues in resource-constrained IoT scenarios, we propose FedASA, a fair and efficient personalized **fed**erated learning scheme with **a**daptive **s**hared **a**rchitecture. On the one hand, shared architectures learn shared features between local devices, which reduces feature drift across devices. On the other hand, reducing the uploading of unnecessary model architectures can significantly reduce the communication overhead. In addition, we perform a theoretical analysis that provides guarantees for the fairness of FedASA.

The contributions of our work are summarized as follows:

- To the best of our knowledge, this is the first work that considers the cross-location issues in resource-constrained IoT scenarios. Our empirical results show that the similarity among data in the cross-location scenario is significantly lower than that in the cross-domain scenario, and the existing federated learning methods suffer from performance degradation in the cross-location scenario.
- We design a fair and efficient personalized FL method called FedASA, which can address the challenge of statistical heterogeneity in resource-constrained IoT scenarios by determining the shared architecture adaptively according to the heterogeneity of devices.
- We provide a theoretical analysis of the federated error bound, which provides the theoretical guarantee for the fairness of FedASA. At the same time, we prove the convergence of FedASA at the first-order stationary point.
- We evaluate the performance of FedASA through extensive simulation and experiments. We demonstrate that FedASA outperforms the state-of-the-art approaches in cross-location scenarios, improving accuracy by up to

13.27% with better fairness and faster convergence, and communication requirement has been reduced by 81.49%.

The rest of this paper is organized as follows. Section II provides motivation and problem formulation. FedASA is proposed in Section III. The theoretical analysis is given in Section IV. Evaluation is presented in Section V. Related works are reviewed in Section VI, followed by the concluding remarks in Section VII.

II. Motivation and Problem Formulation

A. Motivation

According to FL [11], we consider that there exists a parameter server and K local devices cooperating to complete a classification task $\mathcal{T} \triangleq \langle \mathcal{D}, \mathcal{F} \rangle$, where $\mathcal{D}$ is the distribution on $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}$ and $\mathcal{F}$ represents a set of mappings from input space $\mathcal{X}$ to output space $\mathcal{Y}$. Each device owns $D_k = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N_k}$, $D_k \in \mathcal{D}_k$ to complete a subtask $\mathcal{T}_k = \langle \mathcal{D}_k, \mathcal{F}_k \rangle$, where N_k is the number of samples. For device k, the actual loss is $\mathcal{L}_k(\boldsymbol{w}_k) \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_k}[\ell(f_k(\boldsymbol{x};\boldsymbol{w}_k), y)]$ where $f_k \in \mathcal{F}_k$ and $\ell(\cdot)$ is a loss function. In practice, the empirical loss $\hat{\mathcal{L}}_k(\boldsymbol{w}_k) = \frac{1}{N_k}\sum_{i=1}^{N_k}[\ell(f_k(\boldsymbol{x}_i;\boldsymbol{w}_k), y_i)]$ is often used as an estimate of the loss. Therefore, the optimization objective $\boldsymbol{w}^*$ of FL satisfies

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \sum_{k=1}^{K} p_k \mathcal{L}_k(\boldsymbol{w}), \tag{1}$$

where $\boldsymbol{w}_k \in \mathcal{W}_k$ is the model parameter of f_k and p_k is the weighting factor, $p_k \geq 0$, $\sum_{k=1}^{K} p_k = 1$, such as $p_k = \frac{N_k}{N}$, where $N = \sum_{k=1}^{K} N_k$.

Statistical heterogeneity across IoT devices is generated by various factors, including work conditions (e.g., rotating speeds and loads) and monitoring locations. Statistical heterogeneity caused by diverse working conditions across devices has been extensively studied, named the cross-domain issues [14], [15]. In this paper, we refer to the statistical heterogeneity caused by various monitoring locations across devices as the cross-location issues. Note that cross-domain issues are caused by monitored devices (e.g., production machines), and cross-location issues are caused by monitoring devices (e.g., sensors).

We investigate 12 different industrial data from three monitoring locations, including A, B, and C, and four levels of loads, including 0, 1, 2, and 3, in the CWRU dataset [31] as detailed in Section V (numbered from A_0 to C_3). Similar to [16], in the CWRU dataset, we first convert the raw signal into frequency domain data by fast Fourier transform (FFT) (refered to Section III-A). We then roughly evaluate the similarities among 12 different data (referred to Fig. 1) by computing the cosine similarity of the transformed frequency domain data. It is observed that at the same location, the similarities of the data are more than 87.5%, whereas the similarities of the data among different monitoring locations are less than 85%. Among them, locations A and C are the most dissimilar pair, and locations B and C are the most similar pair, but the similarity is still less than 85%. The cross-location issues are demonstrated to bring more severe statistical heterogeneity than cross-domain.

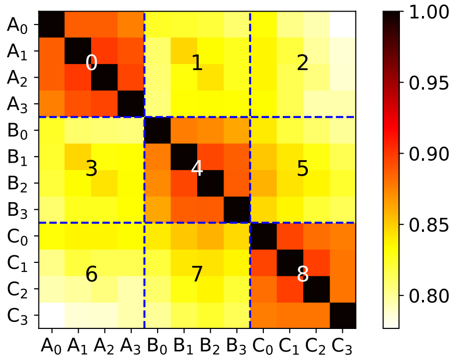


Fig. 1. The cosine similarities among 12 devices with various working conditions and monitoring locations.

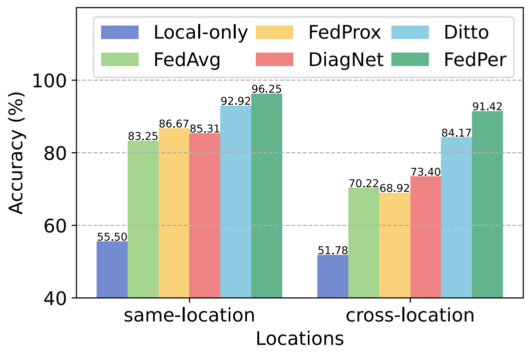


Fig. 2. Performance comparison between same-location and cross-location.

Moreover, Fig. 2 shows the cross-device model accuracy in the same monitoring location (same-location) and different monitoring locations (cross-location). We compared six approaches, including local-only, FedAvg [11], FedProx [27], DiagNet [14]), Ditto [23], FedPer [30]. Experimental results show that these approaches exhibit good performance in cross-domain situations. However, their diagnostic accuracy is reduced in cross-location scenarios. Among them, FedProx shows the largest variation in accuracy, with a drop of 17.75% and FedPer, with the least drop in accuracy of 4.83%. Clearly, it is necessary to solve the cross-location issues.

For statistical heterogeneity, the trained global model $\boldsymbol{w}^*$ cannot meet the personalized requirements of the local device [32], [33]. Therefore, the optimization objective of personalized model $\{\boldsymbol{w}_k^*\}_{k=1}^{K}$ [34] is

$$(\boldsymbol{w}_1^*, \boldsymbol{w}_2^*, \ldots, \boldsymbol{w}_K^*) = \arg\min_{\{\boldsymbol{w}_k\}_{k=1}^{K}} \sum_{k=1}^{K} p_k \mathcal{L}_k(\boldsymbol{w}_k). \tag{2}$$

Note that during the training process of personalized FL, the model parameters of the local device are still uploaded to the server for federated aggregation. Yet, the global optimization is changed to the personalized optimization of each device.

B. Communication Overhead

To deploy an effective and efficient FL framework in real IoT scenarios, we face the problem of resource constrained devices,

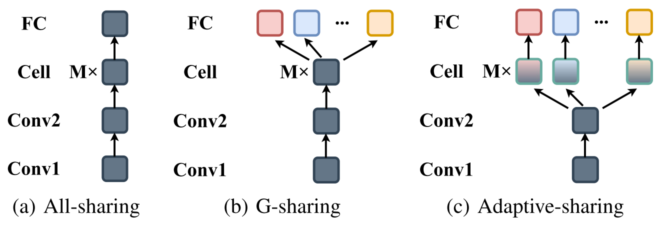


Fig. 3. Illustration of three sharing schemes in federated learning.

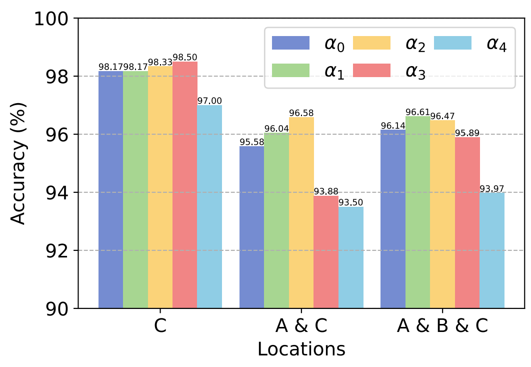


Fig. 4. Performance comparison of different shared layers.

including limited computing and communication resources [20]. Note that the limitation of communication resources is the priority in this paper. Fig. 3 shows three different sharing schemes of the local model in FL. The all-sharing scheme (referred to Fig. 3(a)) shares all layers of the local model, which is used in most FL approaches [11], [14], [23], [27]. Customized model parameters allow these solutions to cater to the specific requirements of different devices and minimize the impact of statistical heterogeneity. Nevertheless, according to the experimental results of Section II-A, they are unsuitable for addressing the cross-location issues in IoT scenarios. In addition, as the local model becomes larger and larger, the communication overhead brought by the all-sharing scheme cannot be ignored. The G-sharing scheme (as shown in Fig. 3(b)) shares only the lowest G layers of the local model. Typically, G is predetermined and fixed in the training process. This baseline is similar to several existing personalized FL approaches [30], [35]. The G-sharing scheme is indeed able to reduce the communication overhead, even that introduced by a fully connected layer. According to the experimental results of Section II-A, G-sharing has the prospect of adapting to the cross-location issues. However, G is crucial for the local device, and determining the optimal G brings additional computation and communication overhead. On the other hand, due to the effects of statistical heterogeneity, especially for cross-location issues, a fixed upload architecture for all devices may be a suboptimal solution.

To further exam our idea, we conducted a simple experiment in CWRU dataset. As shown in the Fig. 4, α_0 to α_4 indicates that the number of shared layers is increasing. Experimental results show that when the data only comes from location C, the optimal shared layers is α_3, which means more shared layers, and as monitoring locations increase, the number of shared layers required by devices decreases. In more complex heterogeneous scenarios, reducing the requirement for shared layers may be possible. Therefore, heterogeneous shared architectures are required, and it is necessary to design shared architectures adaptively according to the heterogeneity of devices. As shown in Fig. 3(c), our goal is to propose an adaptive-sharing scheme, which adaptively determines the shared architecture for each device according to the dataset of the local device.

In general, (2) can be solved by gradient descent algorithms [11], such as mini-batch gradient descent (mini-batch SGD). For the mini-batch SGD, an iteration represents that the local device performs gradient descent on a mini-batch of data. One round of local updates consists of multiple iterations. After a round of local updates, the local device pushes the local model $\boldsymbol{w}_k^{t+1} = \boldsymbol{w}_k^t - \eta \nabla \mathcal{L}_k(\boldsymbol{w}^t)$ (η is the learning rate) to the server for aggregation.

Note that the communication overhead of FL is determined by the size of transmitted model parameters [36]. Assuming that in round t, for device k, the bandwidth requirement $\mathcal{C}_k^t$ can be formulated by

$$\mathcal{C}_k^t = \mathcal{CO}(\boldsymbol{w}_k^t), \tag{3}$$

where the size of the model parameters $\boldsymbol{w}_k^t$ is directly proportional to $\mathcal{CO}(\cdot)$. FedASA reduces the communication overhead by reducing the size of the transmitted model parameters.

C. Problem Formulation

Statistical heterogeneity can lead to significant variation in model performance across devices. From the perspective of profit, the local device cares more about the performance of the private model than the average performance of all devices or the best. Therefore, more attention should be paid to fairness among devices while improving the average performance. Following [23], [37], the fairness of the model can be defined as follows

Definition 1 (Fairness): When comparing the fairness of the two models, we consider that the fairer one has more uniform performance across devices, i.e., for $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, if $std(\{\mathcal{L}_k(\boldsymbol{w}_1)\}_{k=1}^K) < std(\{\mathcal{L}_k(\boldsymbol{w}_2)\}_{k=1}^K)$, where $std(\cdot)$ is the standard deviation, then $\boldsymbol{w}_1$ is fairer than $\boldsymbol{w}_2$.

FedASA ensures that devices have similar performance levels, reducing the impact of fairness barriers. To realize personalization, FedASA divides the local model into shared and personalized architecture. Let $f_k = g_k \circ h_k$, where $g_k \in \mathcal{G}_k$ represents the shared architecture, $h_k \in \mathcal{H}_k$ is the personalized architecture, and the parameters of g_k is $\boldsymbol{\theta}_k \in \Theta_k$, and the parameters of h_k is $\boldsymbol{\phi}_k \in \Phi_k$. Therefore, our optimization objective is

$$(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) = \underset{\{\boldsymbol{\theta}_k, \boldsymbol{\phi}_k\}_{k=1}^K}{\arg\min} \sum_{k=1}^{K} p_k \mathcal{L}_k(\boldsymbol{\theta}_k \circ \boldsymbol{\phi}_k), \tag{4}$$

where $\boldsymbol{\theta}^* = (\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*, \ldots, \boldsymbol{\theta}_K^*)$, $\boldsymbol{\phi}^* = (\boldsymbol{\phi}_1^*, \boldsymbol{\phi}_2^*, \ldots, \boldsymbol{\phi}_K^*)$. Similarly, (3) is rewritten as

$$\mathcal{C}_k^t = \mathcal{CO}(\boldsymbol{\theta}_k^t). \tag{5}$$

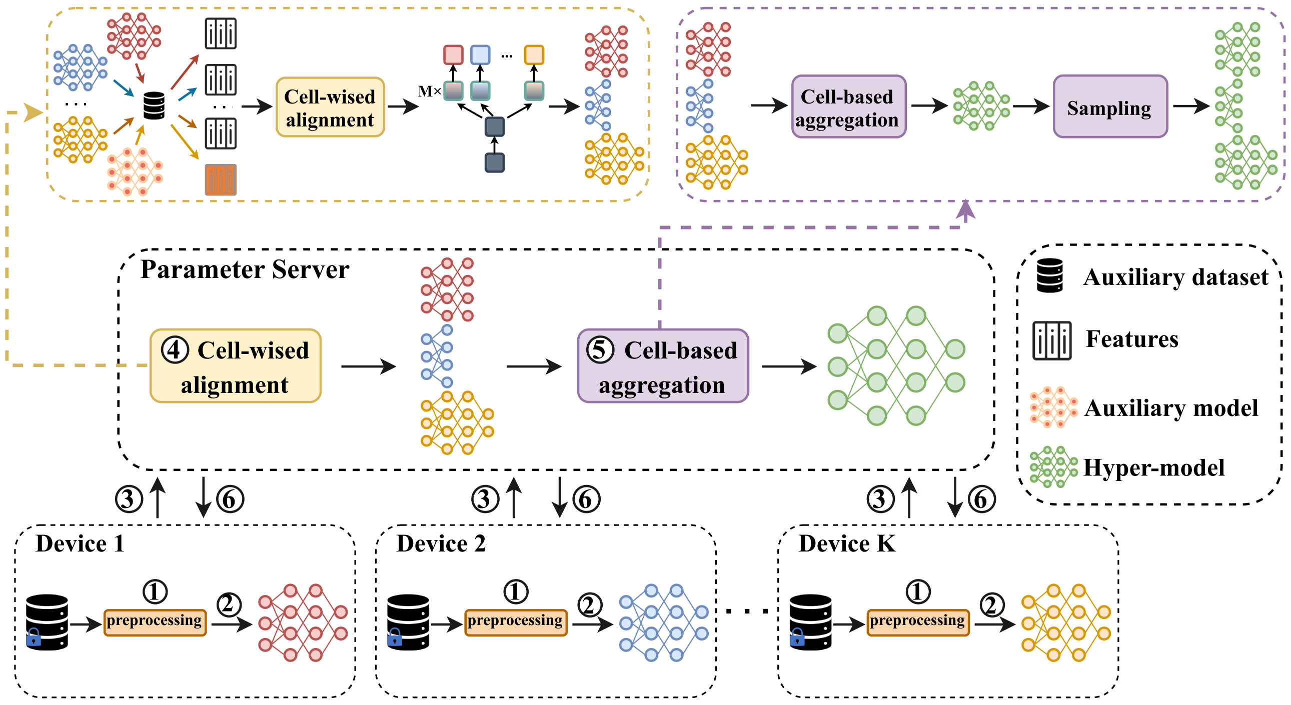


Fig. 5. Overview of FedASA. The training procedure consists of the following steps: ①. Data preprocessing. ②. Local training on private data. ③. Each device uploads the trained model parameters to the parameter server. ④. The parameter server adaptively determines the global shared architectures by cell-wised alignment. ⑤. The parameter server updates the hyper-model by cell-based aggregation and sample the trained local model parameters. ⑥. Each device downloads the updated model parameters. Repeat ②-⑥ until the model converges or meets the accuracy requirements.

TABLE II
A SUMMARY OF IMPORTANT NOTATIONS

Notation	Description
$\mathcal{D}$	Distribution on $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}$
$\mathcal{F}$	Set of mappings from input space $\mathcal{X}$ to output space $\mathcal{Y}$
D_k	Dataset for device k
$\boldsymbol{w}$	Global model
$\boldsymbol{w}_k$	Local model of device k
$\boldsymbol{\theta}_k$	Shared architecture of device k
$\boldsymbol{\phi}_k$	Personalized architecture of device k
p_k	Weighting factor
$\boldsymbol{\theta}_{hp}(\cdot)$	Hyper-model
$\mathcal{L}(\cdot)$	Global loss
$\hat{\mathcal{L}}(\cdot)$	Empirical loss
K	Total number of devices
T	Total number of communication rounds
$\mathcal{C}_k$	Bandwidth resource budget of device k
$\mathcal{C}_k^t$	Bandwidth consumption of device k in t-th round
η	Learning rate

Assuming that for device k, the total communication resource budget is $\mathcal{C}_k$. Accordingly, we formulate the problem as follows:

$$\min_{\boldsymbol{\theta}_k,\boldsymbol{\phi}_k} \sum_{k=1}^{K} p_k \mathcal{L}_k(\boldsymbol{\theta}_k \circ \boldsymbol{\phi}_k),$$

$$\text{s.t.} \begin{cases} \mathcal{L}_k(\boldsymbol{\theta}_k^T \circ \boldsymbol{\phi}_k^T) - \mathcal{L}_k(\boldsymbol{\theta}_k^* \circ \boldsymbol{\phi}_k^*) < \varepsilon_1, \\ \mathcal{C}_k^t \le \mathcal{C}_k, \\ std(\{\mathcal{L}_k(\boldsymbol{\theta}_k^T \circ \boldsymbol{\phi}_k^T)\}) < \varepsilon_2, \\ \forall k \in [K], \end{cases} \tag{6}$$

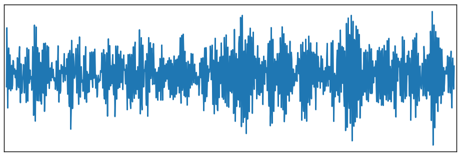
where $\varepsilon_1, \varepsilon_2 > 0$ are the predefined threshold (close to zero) and we use $k \in [K]$ represent $k \in \{0, 1, \ldots, K\}$. Some important notations in this paper are listed in Table II.

III. Personalized Federated Learning With Adaptive Shared Architecture

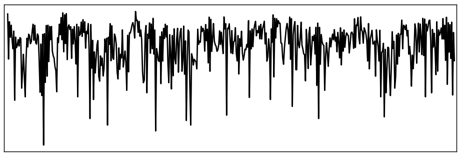
As illustrated in Fig. 5, we present an overview of FedASA. It consists of two main steps: data preprocessing and model training. Specifically, the workflow of FedASA is as follows: (i) The server initializes the global model parameters and local device training epochs E, which are broadcast to all devices. Each device performs data preprocessing. (ii) After local training, each device sends the trained model parameters to the parameter server. (iii) The server uses cell-wised alignment to identify the shared architectures and then updates the hyper-model through cell-based aggregation. (iv) The server broadcasts the sampled local device parameters to each device. (v) Repeat ii-iv until the model converges or meets the accuracy requirements. In the following, we first give the data preprocessing of FedASA.

A. Data Preprocessing

In IoT scenarios, it is common practice to train a model directly using sensor signals to reduce the dependence on expert knowledge [16], [17]. Note that we analyze the case where local labels are uniformly distributed. We exclude the case where the local labels are non-uniformly distributed from our considerations. Statistical heterogeneity across devices only comes from cross-domain and cross-location issues. For each



(a) Raw signal



(b) Frequency domain signal

Fig. 6. The original time domain signal and frequency domain signal comparison.

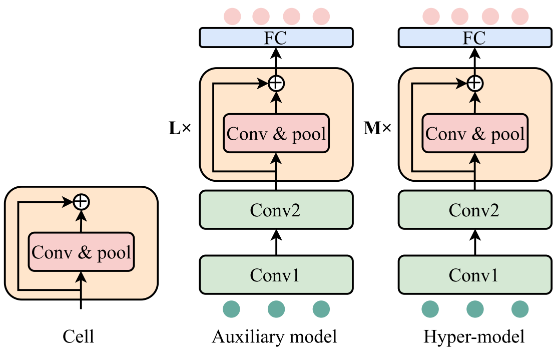


Fig. 7. Architectures of different models.

device, the private dataset contains all types of signals to build a multi-classification task and each signal is divided into samples of length 1024.

Since the raw vibration signals only represent the time domain (referred to Fig. 6(a)) and their phases are different, the cosine similarities among the devices are very low, less than 3% in CWRU dataset. In Section II-A, the FFT is used to convert raw signals to obtain frequency domain signals (referred to Fig. 6(b)). The FFT algorithm is used for calculating the discrete Fourier transform (DFT) of a sequence. The DFT is obtained by decomposing a series of values into components of different frequencies. Let $\boldsymbol{x} = (x_0, \dots, x_{S-1})$ be a signal. The DFT is defined by (7),

$$X_d = \sum_{s=0}^{S-1} x_s e^{-i2\pi ds/S}, \quad d = 0, \dots, S-1, \tag{7}$$

where $e^{i2\pi/S}$ is a primitive S-th root of 1.

B. Cell-Wised Alignment

To facilitate training, FedASA simplifies the model architecture to a cell-based architecture. As shown in Fig. 7, the residual-based cell comprises a convolutional layer followed by a pooling layer, with equal input and output feature dimensions. Each local model contains two convolutional layers, M cells and FC layers, identical to the hyper-model architecture shown in Fig. 7. This design has two advantages. Firstly, it ensures that the feature dimension of the output of each cell is the same, which makes it simpler to divide heterogeneous architectures. Secondly, due to the residual structure, it prevents the degradation of deep network models. In order to determine the adaptive model architecture, the auxiliary model trained from the parameter server's auxiliary dataset is essential. In IoT scenarios, collecting an auxiliary dataset is a simple task because each batch of devices is tested before being sold. In our experience, the auxiliary data can come from only one monitoring location and the trained auxiliary model can perform poorly with only 78.12%. Note that the auxiliary model and hyper-model have similar architectures. The auxiliary model contains L cells, whereas the hyper-model contains M cells ($L \le M$). Since the shared architectures across devices are heterogeneous, we introduce a hyper-model to assist aggregation. Specific details for aggregation can be found in the cell-based aggregation algorithm. Fig. 7 displays the architecture of these models.

Note that FedASA adaptively determines the shared architecture g_k based on the private dataset in each device. In the real-world FL scenarios, achieving this goal is challenging. Fortunately, the auxiliary dataset D_{aux} and auxiliary model g_{aux} of the parameter server are available. To address this challenge, we propose a cell-wised alignment algorithm, which takes the output features of the cells as the criterion. After uploading all local models, they are processed by the auxiliary dataset to capture output features from each cell. The auxiliary model's output features are aligned with the client model's. Note that the optimal sharing architecture must satisfy both accuracy and communication resource requirements. Directly selecting the minimum alignment distance may be biased, which may cause additional communication overhead for the local device. Let the auxiliary model g_{aux}^{ℓ} contain ℓ cells, and the local model g_k^m contain m cells, then the optimal alignment cell m_k^* is

$$\begin{aligned} m_k^* &= \underset{m \in [M]}{\arg\min}\, \mathbb{E}_{\boldsymbol{x} \in D_{aux}} \| g_{aux}^{\ell}(\boldsymbol{x}; \boldsymbol{\theta}_{aux}) - g_k^m(\boldsymbol{x}; \boldsymbol{\theta}_k) \|^2, \\ &\text{s.t.} \quad \ell \in [L],\ \mathcal{C}_k^t \le \mathcal{C}_k, \end{aligned} \tag{8}$$

where ℓ is a hyperparameter. Furthermore, $\boldsymbol{m}^* = (m_1^*, m_2^*, \dots, m_K^*)$ is referred to as the alignment vector, which is used to record the shared architecture in the next round.

C. Cell-Based Aggregation

Statistical heterogeneity can cause shared architectures to have varying numbers of cells, which are stochastically exposed to the datasets of devices. As such, naively using FedAvg [11] for aggregation is not feasible since it is designed to aggregate the same architecture. For this reason, we propose a cell-based aggregation algorithm, where the update experience of the cell determines the weight of the update. Let $\mathcal{O} = \{o_0, o_1, o_2, \dots, o_M\}$ be the set of operators involved in aggregation, e.g. the convolutional layer and cell. Let $\boldsymbol{\theta}(o_0)$ be the parameters of the two convolutional layers and $\boldsymbol{\theta}(o_m)$ be the parameters of the m-th

```
Algorithm 1: The FedASA Algorithm.
Input: Communication resources budget for each device $\mathcal{C}_k$,
    training round $T$, auxiliary dataset $D_{aux}$, auxiliary model
    $g_{aux}$, alignment rounds $t_0$
Output: $\boldsymbol{\theta}_k^T$, $\boldsymbol{\phi}_k^T$
 1: Initialize the devices' model parameters $\boldsymbol{\theta}_k^0$, $\boldsymbol{\phi}_k^0$,
    hyper-model parameters $\boldsymbol{\theta}_{hp}^0(\cdot)$
 2: At Parameter Server:
 3: for $t = 1, 2, \dots, T$ do
 4:     if $t \leq t_0$ then
 5:         Receive $\boldsymbol{w}_k^t$ for all $k \in [K]$
 6:         $m_k^* = \arg\min_{m \in [M]} \mathbb{E}\|g_{aux}^{\ell}(\boldsymbol{x}; \boldsymbol{\theta}_{aux}) - g_k^m(\boldsymbol{x}; \boldsymbol{\theta}_k)\|^2$ //Refer to (8)
 7:         Collect the alignment vector $\boldsymbol{m}^*$
 8:     else
 9:         Receive $\boldsymbol{\theta}_k^t$ for all $k \in [K]$
10:     end if
11:     if $S(o) 0$ then
12:         $\boldsymbol{\theta}_{hp}^{t+1}(o) = \sum_{k=1}^{K} \frac{N_{k,o}^t}{\sum_{i=1}^{K} N_{i,o}^t} \boldsymbol{\theta}_k^t(o)$ //Refer to (9)
13:     else
14:         $\boldsymbol{\theta}_{hp}^{t+1}(o) = \boldsymbol{\theta}_{hp}^t(o)$ //Refer to (9)
15:     end if
16:     Sample updated shared models $\boldsymbol{\theta}^{t+1}$
17:     Broadcast $\boldsymbol{\theta}^{t+1}$ to all devices
18: end for
19: At Device $k$:
20: for $t = 1, 2, \dots, T$ do
21:     Receive $\boldsymbol{\theta}_k^t$ from the Parameter Server
22:     $\boldsymbol{\theta}_k^{t+1} = \boldsymbol{\theta}_k^t - \eta_g \nabla \hat{\mathcal{L}}_k(\boldsymbol{\theta}_k \circ \boldsymbol{\phi}_k)$ //Refer to (13)
23:     $\boldsymbol{\phi}_k^{t+1} = \boldsymbol{\phi}_k^t - \eta_h \nabla \hat{\mathcal{L}}_k(\boldsymbol{\theta}_k \circ \boldsymbol{\phi}_k)$ //Refer to (13)
24:     Send $\boldsymbol{\theta}_k^{t+1}$ to the Parameter Server
25: end for
```

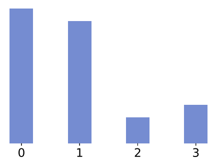
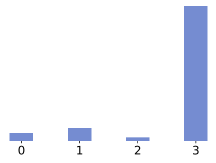
cell. For all $o \in \mathcal{O}$,

$$\boldsymbol{\theta}_{hp}^{t+1}(o) = \begin{cases} \sum_{k=1}^{K} \frac{N_{k,o}^t}{\sum_{i=1}^{K} N_{i,o}^t} \boldsymbol{\theta}_k^t(o), & S(o) > 0, \\ \boldsymbol{\theta}_{hp}^t(o), & \text{otherwise}, \end{cases} \tag{9}$$

where $\boldsymbol{\theta}_{hp}^t(\cdot)$ are the parameters of hyper-model in round t, $\boldsymbol{\theta}_k^t(\cdot)$ are parameters of device k in round t, $N_{k,o}^t$ is the number of samples using operator o in round t, and $S(o)$ denotes the frequency of using the operator o during a round of updates. An operator in the hyper-model is updated only if it has been used more than once. Finally, if an operator (such as operator o_0) is used in all devices, then (9) degenerates into FedAvg. Specifically, FedASA is a generalization of FedAvg, where the average is performed independently for each operator rather than for the full architecture.

Sampling: After the hypermodel update, the parameter server has completed one round of updates. Delivering the hyper-model directly is inappropriate as each device only needs the same model parameters as the shared architecture, which incurs additional communication overhead. We can sample the updated shared architectures of devices by using an alignment vector $\boldsymbol{m}^* = (m_1^*, m_2^*, \dots, m_K^*)$.

TABLE III
Two Examples

Example	soft label	hard label	GT	cross-entropy	$\hat{\mathcal{L}}_k(\cdot)$
1		0	1	1.23	2.42
2		3	3	0.85	1.42

Example 1 shows low confidence and Example 2 shows high confidence. GT stands for the groundtruth label.

D. Algorithm Description

Although classification tasks with cross-entropy would intuitively bring improvement, unfortunately, we cannot directly access supervision information of the target tasks [12]. For a given sample x_i^k from device k, we have

$$s_i^k = \mathrm{softmax}(h(g(x_i^k))), \tag{10}$$

where s_i^k is a soft label of x_i^k. Meanwhile, a hard label for x_i^k is generated by one-hot encoder

$$h_i^k = \mathrm{one\_hot}(\max(s_i^k)). \tag{11}$$

If the model prediction vectors with high confidence, soft labels may be more useful for model updating. If the prediction is not confident enough, the hard labels can be used to optimize the model update. High and low confidence examples are shown in Table III. Replacing with the hard label at the proper time can improve the model [38]. To balance the soft label and hard label, we define the following loss function:

$$\begin{aligned} &\hat{\mathcal{L}}_k(\boldsymbol{\theta}_k \circ \boldsymbol{\phi}_k) \\ &= -\frac{1}{N_k} \sum_{c=1}^{C} \mathbb{1}\{y_i = c\} \left(\log \frac{e^{s_{i,c}^k}}{\sum_{j=1}^{C} e^{s_{j,c}^k}} - e^{\mathcal{E}} \right), \end{aligned} \tag{12}$$

where $\mathcal{E} = -\sum_{i=1}^{C} s_{i,c}^k \log s_{i,c}^k$ is the entropy of $s_{i,c}^k$ and C is the number of categories.

During local training, $\boldsymbol{\phi}_k$ and $\boldsymbol{\theta}_k$ of the model are merged to $\boldsymbol{w}_k$. Assuming that the gradient descent algorithm is used for model optimization, then for device k,

$$\begin{cases} \boldsymbol{\theta}_k^{t+1} = \boldsymbol{\theta}_k^t - \eta_g \nabla \hat{\mathcal{L}}_k(\boldsymbol{\theta}_k \circ \boldsymbol{\phi}_k), \\ \boldsymbol{\phi}_k^{t+1} = \boldsymbol{\phi}_k^t - \eta_h \nabla \hat{\mathcal{L}}_k(\boldsymbol{\theta}_k \circ \boldsymbol{\phi}_k), \end{cases} \tag{13}$$

where η_g is the learning rate of $\boldsymbol{\theta}_k$ and η_h is the learning rate of $\boldsymbol{\phi}_k$.

To sum up, for device k, the algorithm for each round of updates is: (i) Initialize the devices' model parameters $\boldsymbol{\theta}_k^0$, $\boldsymbol{\phi}_k^0$ or receive the updated $\boldsymbol{\phi}_k^t$. (ii) $\boldsymbol{\phi}_k^t$ and $\boldsymbol{\theta}_k^t$ are trained locally for E epochs regarding (13). (iii) Send the trained $\boldsymbol{\theta}_k^{t+1}$ to the parameter server. For the server, the algorithm for each round of updates is: (i) Receive all updated $\boldsymbol{\theta}_k^t$. (ii) If $t = 1$ perform to determine the optimal $\boldsymbol{m}^*$ by cell-wised alignment otherwise skip this step. (iii) Update hyper-model $\boldsymbol{\theta}_{hp}^{t+1}(\cdot)$ through cell-based aggregation. (iv) Sample updated shared models $\boldsymbol{\theta}_k^{t+1}$ by $\boldsymbol{m}^*$. (v) Send $\boldsymbol{\theta}_k^{t+1}$ to all devices. Follow this loop for multiple rounds of training until the model converges or reaches the accuracy requirements. The detailed algorithm is referred to as Algorithm 1. Algorithm 1 begins with estimating the communication resources budget for each device $\mathcal{C}_k$, the setting of training round T, and the collection of auxiliary dataset D_{aux} and auxiliary model g_{aux}. Line 1 is the initialization of the local model parameters $\boldsymbol{\theta}_k^0$, $\boldsymbol{\phi}_k^0$ and hyper-model parameters $\boldsymbol{\theta}_{hp}^0(\cdot)$. At the parameter server, lines 4-8 are used to perform cell-wised alignment and to collect the alignment vectors $\boldsymbol{m}^*$. Lines 11-16 are used to perform cell-based aggregation and sample updated shared models $\boldsymbol{\theta}^{t+1}$. Line 17 is used to broadcast $\boldsymbol{\theta}^{t+1}$ to all devices. At device k, line 21 is used to receive the updated $\boldsymbol{\theta}^t$. Lines 22-23 are used to update the local model parameters $\boldsymbol{\theta}_k^{t+1}$, $\boldsymbol{\phi}_k^{t+1}$. Line 24 is used to send $\boldsymbol{\theta}_k^{t+1}$ to the parameter server.

IV. Theoretical Analysis

FedASA divides local model architectures into shared and personalized architectures, which is similar to domain adaptation. According to the theory of federated domain adaptation [34], [39], we performed a theoretical analysis of FedASA. In this section, we first analyze the federal error bound of FedASA, and then provide the convergence analysis. Note that error is a statistical object, and we use loss to quantify the negative effect of error.

A. Federated Error Bound

In supervised learning, given a set of training samples, the goal is to find the optimal function $f^* \in \mathcal{F}$ by minimizing the error on the test set. However, this relies on the assumption that the training and test sets are independent and identically distributed. In actual production, this assumption requires a lot of cost for data preprocessing. Domain adaptation aims to solve the problem of inconsistent data distribution.

In typical domain adaptation, we are referring to a labeled source domain $\mathcal{D}_1$ and an unlabeled target domain $\mathcal{D}_2$. The objective is to find a function $f \in \mathcal{F}$ that minimizes the generalization error $\mathcal{L}_2 = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_2}[\ell(f(\boldsymbol{x};\boldsymbol{w}), y)]$ of the target domain. In [40], with a probability of at least $1-\delta$, we get the following error upper bound

$$\mathcal{L}_2(f) \leq \mathcal{L}_1(f) + \underbrace{\frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{D}_1, \mathcal{D}_2)}_{\text{Ideal Joint Error}} + \underbrace{4\sqrt{\frac{2d\log(2m) + \log\frac{2}{\delta}}{m}} + \lambda}_{\text{Disparity Difference}}, \tag{14}$$

where $\delta \in (0,1)$, $\lambda = \mathcal{L}_1(f^*) + \mathcal{L}_2(f^*)$ is a constant, and for $f, f' \in \mathcal{F}$, $\oplus$ is the XOR function, then $\mathcal{F}\Delta\mathcal{F}$ is equivalent to $f(\boldsymbol{x}) \oplus f'(\boldsymbol{x})$. (14) shows that the target domain error is related not only to the source domain error but also to the ideal joint error and disparity difference. Subsequent domain adaptation theorems, such as [34], [39], are based on (14).

FL involves a series of independent and identically distributed devices, which differs from classical domain adaptation. Multi-source domain adaptation faces the following challenges: (i) There are multiple devices, and data sharing is limited among each other. (ii) The adaptation effects can vary among devices due to statistical heterogeneity. Fortunately, in FL, a powerful parameter server is available. In FL, K devices work together to train a generalization model that can adapt to device i. Similarly, ensemble learning (EL) aims to create a generalization model that can adapt to the new device, e.g., a weighted ensemble. But which one is better? In this brief analysis, we will compare them in terms of error bounds. According to the above analysis, the target domain error is related not only to the source domain error but also to ideal joint error and disparity difference. Obviously, FL and EL are similar in the latter two parts, so their source domain errors are considered. In FL, the source domain error is $\mathcal{L}_{FL} = \mathcal{L}(\sum_{k=1}^K p_k f_k)$, while in EL, it is $\mathcal{L}_{EL} = \sum_{k=1}^K p_k \mathcal{L}(f_k)$. If the loss function is convex, then it is obvious $\mathcal{L}_{FL} \leq \mathcal{L}_{EL}$. However, the loss function $\mathcal{L}(\cdot)$ often satisfies L-smooth (referred to Definition 2). For L-smooth functions, the lower bound of $\mathcal{L}(\cdot)$ is a hyperplane, and its upper bound is a quadratic function. Its upper bound is convex, and its lower bound also satisfies $\mathcal{L}_{FL} \leq \mathcal{L}_{EL}$. So, we can get $\mathcal{L}_{FL} \leq \mathcal{L}_{EL}$, which means FL achieves a minor error bound when obtaining the objective function f^*.

Next, we provide various federated error bounds. According to Theorem 2 of [41], let $\tilde{\mathcal{D}} = \sum_{k=1}^K p_k \mathcal{D}_k$ be the mixture distribution of all devices, and the corresponding mixture error is $\tilde{\mathcal{L}}(\cdot)$. With a probability of at least $1-\delta$, the following error upper bound can be obtained:

$$\mathcal{L}_i(f) \leq \tilde{\mathcal{L}}\left(\sum_{k=1}^K p_k f_k\right) + \sum_{k=1}^K p_k \left(\frac{1}{2}\hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{D}_k, \mathcal{D}_i) + \lambda_k\right) + 4\sqrt{\frac{2d\log(2Km) + \log\frac{2}{\delta}}{Km}}, \tag{15}$$

where $\lambda_k = \mathcal{L}_k(f^*) + \mathcal{L}_i(f^*)$.

The loss upper bound (15) is only suitable for all-sharing scenarios. In [34], Regatti et al. divided the local model into the base layer u_k and the top layer v_k. The globally optimal v_k is defined as $\hat{v} = \arg\min_v \mathcal{L}_{\tilde{\mathcal{D}}}(\cdot)$. The local optimal v_k is defined as $v_k^* = \arg\min_v \mathcal{L}_{\mathcal{D}_k}(\cdot)$. With a probability of at least $1-\delta$, the following loss upper bound can be obtained:

$$\mathcal{L}_k(\hat{v}) - \mathcal{L}_k(v_k^*) \leq \frac{1}{2}\sum_{i=1}^K p_i k \hat{d}_{\mathcal{F}\Delta\mathcal{F}}(\mathcal{D}_i, \mathcal{D}_k) + |\hat{\lambda}_v - \lambda_v^*| \tag{16}$$

where $\hat{\lambda}_v$ and λ_v^* represent the error constants when v_k is global optimal and local optimal, respectively. The loss upper bound (16) focuses on the top layer v_k, but we focus more on the shared architecture g_k. For device k, g_k is sampled from the hyper-model g. Let the global optimal model g^* satisfy $\tilde{\mathcal{L}}(g^*) = \sum_{k=1}^K p_k \mathcal{L}_k(g_k^*)$, and then we have the following theorem:

Theorem 1: Let the VC-dimension of the function space $\mathcal{G}$ be d and the number of local samples be m. The estimate of the disparity difference between the two devices is $\hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\mathcal{D}_i, \mathcal{D}_j)$. For any $\delta \in (0,1)$ and $g_k \in \mathcal{G}$, with a probability of at least $1-\delta$, we have

$$\mathcal{L}_i(g_i) \le \tilde{\mathcal{L}}(g) + \underbrace{\frac{1}{2}\sum_{k=1}^K p_k \hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\mathcal{D}_k, \mathcal{D}_i)}_{\text{Disparity Difference}} + \underbrace{4\sqrt{\frac{2d\log(2Km) + \log\frac{2}{\delta}}{Km}} + \lambda^*}_{\text{Ideal Joint Error}}, \tag{17}$$

where $\lambda^* = \mathcal{L}(g^*) + \mathcal{L}_i(g^*)$.

Proof: According to (15), we can get

$$\begin{aligned}\mathcal{L}_i(g_i) &\le \tilde{\mathcal{L}}\left(\sum_{k=1}^K p_k g_k\right) + \sum_{k=1}^K p_k\left(\frac{1}{2}\hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\mathcal{D}_k, \mathcal{D}_i) + \lambda_k\right) \\ &\quad + 4\sqrt{\frac{2d\log(2Km) + \log\frac{2}{\delta}}{Km}} \\ &= \tilde{\mathcal{L}}(g) + \frac{1}{2}\sum_{k=1}^K p_k \hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\mathcal{D}_k, \mathcal{D}_i) \\ &\quad + 4\sqrt{\frac{2d\log(2Km) + \log\frac{2}{\delta}}{Km}} + \sum_{k=1}^K p_k\lambda_k \end{aligned} \tag{18}$$

Moreover, if the global optimal model g^* satisfies $\mathcal{L}(g^*) = \sum_{k=1}^K p_k \mathcal{L}_k(g_k^*)$, and also define $\lambda^* = \mathcal{L}(g^*) + \mathcal{L}_i(g^*)$, then (18) can be rewritten as:

$$\mathcal{L}_i(g_i) \le \tilde{\mathcal{L}}(g) + \underbrace{\frac{1}{2}\sum_{k=1}^K p_k \hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\mathcal{D}_k, \mathcal{D}_i)}_{\text{Disparity Difference}} + \underbrace{4\sqrt{\frac{2d\log(2Km) + \log\frac{2}{\delta}}{Km}} + \lambda^*}_{\text{Ideal Joint Error}} \tag{19}$$

■

Note that the error bound presented in Theorem 1 is a generalized version of (15). In FedASA, the cell-wised alignment scheme makes the extracted features among g_k exhibit high similarity. Clearly, compared with (15), the error bound of FedASA is minor, which means that the standard deviation of loss across devices is minor. According to Definition 1, FedASA offers better fairness.

B. Convergence Analysis

Let $\boldsymbol{w}$ denote the concatenation of $(\boldsymbol{\theta}, \boldsymbol{\phi})$. First, we introduce related definitions and assumptions.

Definition 2: A differentiable function $f(x)$ is called **L-smooth**, iff it has a Lipschitz continuous gradient, i.e., iff $\exists L < \infty$ such that

$$\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\| \le L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \ \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d. \tag{20}$$

Assumption 1: The global loss function $\mathcal{L}(\cdot)$ satisfies L-smooth, i.e., $\forall \boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$,

$$\mathcal{L}(\boldsymbol{w}_1) - \mathcal{L}(\boldsymbol{w}_2) \le \langle \nabla\mathcal{L}(\boldsymbol{w}_2), \boldsymbol{w}_1 - \boldsymbol{w}_2\rangle + \frac{L}{2}\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|^2. \tag{21}$$

Assumption 2: The expected squared norm of stochastic gradients is uniformly bounded, i.e., $\forall k \in [K], \forall t \in [T], \exists G$, such that

$$\mathbb{E}\|\nabla\hat{\mathcal{L}}_k(\boldsymbol{w}^t)\|^2 \le G^2. \tag{22}$$

Assumption 3: The gradient $\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t) = \sum_{k=1}^K p_k \nabla\hat{\mathcal{L}}_k(\boldsymbol{w}^t)$ is bounded, i.e., $\exists a0$ such that

$$\nabla\mathcal{L}(\boldsymbol{w}^t)^T \mathbb{E}(\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)) \ge a\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2. \tag{23}$$

Assumption 1 is commonly present in multiple federated learning approaches, including FedProx [27] and FedCMA [34]. Assumption 2 restricts the gradient's scope during the update. Assumption 3 ensures that $\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)$ is an estimate of $\nabla\mathcal{L}(\boldsymbol{w}^t)$. If and only if $a = 1$, we have $\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)$ as the unbiased estimation of $\nabla\mathcal{L}(\boldsymbol{w}^t)$.

We then introduce several useful lemmas to enhance our comprehension of the theory.

Lemma 1: With Assumption 2, for aggregated model $\hat{\mathcal{L}}(\boldsymbol{w}^t)$, we have,

$$\mathbb{E}\|\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)\|^2 \le K^2G^2 \tag{24}$$

Proof: Using $\nabla\hat{\mathcal{L}}(\boldsymbol{w}) = \sum_{k=1}^K p_k \nabla\hat{\mathcal{L}}_{\mathcal{D}_k}(\boldsymbol{w}_k)$ and Assumption 3, we can get

$$\begin{aligned}\|\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)\|^2 &= \left\|\sum_{k=1}^K p_k\nabla\hat{\mathcal{L}}_k(\boldsymbol{w}^t)\right\|^2 \\ &\le \|\nabla\hat{\mathcal{L}}_1(\boldsymbol{w}^t) + \nabla\hat{\mathcal{L}}_2(\boldsymbol{w}^t) + \cdots + \nabla\hat{\mathcal{L}}_k(\boldsymbol{w}^t)\|^2 \\ &\le K^2\|\nabla\hat{\mathcal{L}}_{k^*}(\boldsymbol{w}^t)\|^2 \\ &\quad \left(k^* = \arg\max_k \nabla\hat{\mathcal{L}}_k(\boldsymbol{w}^t)\right)\end{aligned} \tag{25}$$

Take expectation of both sides,

$$\begin{aligned}\mathbb{E}\|\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)\|^2 &\le K^2\mathbb{E}\|\nabla\hat{\mathcal{L}}_{k^*}(\boldsymbol{w}^t)\|^2 \\ &\le K^2G^2\end{aligned} \tag{26}$$

■

Lemma 2: If $\mathcal{L}(\cdot)$ is L-smooth, then with Assumptions 1 and 3, we have,

$$\mathbb{E}[\mathcal{L}(\boldsymbol{w}^{t+1})] \le \mathbb{E}[\mathcal{L}(\boldsymbol{w}^t)] - \eta a\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2 + \frac{1}{2}L\eta^2K^2G^2 \tag{27}$$

Proof: Using the smoothness property of $\mathcal{L}(\cdot)$ from Assumption 1, we can get

$$\mathcal{L}(\boldsymbol{w}^{t+1}) \le \mathcal{L}(\boldsymbol{w}^t) + \langle\nabla\mathcal{L}(\boldsymbol{w}^t), \boldsymbol{w}^{t+1} - \boldsymbol{w}^t\rangle$$

$$
\begin{aligned}
&+\frac{L}{2}\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^{t}\|^2\\
&=\mathcal{L}(\boldsymbol{w}^t)-\eta\nabla\mathcal{L}(\boldsymbol{w}^t)^T\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)\\
&+\frac{L\eta^2}{2}\|\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)\|^2 \qquad (28)
\end{aligned}
$$

Take expectation of both sides,

$$
\begin{aligned}
\mathbb{E}[\mathcal{L}(\boldsymbol{w}^{t+1})] &\le \mathbb{E}[\mathcal{L}(\boldsymbol{w}^t)]-\eta\nabla\mathcal{L}(\boldsymbol{w}^t)^T\mathbb{E}[\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)]\\
&+\frac{L\eta^2}{2}\mathbb{E}\|\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)\|^2\\
&\le \mathbb{E}[\mathcal{L}(\boldsymbol{w}^t)]-\eta a\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2\\
&+\frac{L\eta^2}{2}\mathbb{E}\|\nabla\hat{\mathcal{L}}(\boldsymbol{w}^t)\|^2\\
&\le \mathbb{E}[\mathcal{L}(\boldsymbol{w}^t)]-\eta a\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2\\
&+\frac{1}{2}L\eta^2K^2G^2 \qquad (29)
\end{aligned}
$$

■

Theorem 2: Let Assumptions 1, 2 and 3 hold and after running the algorithm with T rounds, we have

$$
\sum_{t=0}^{T-1}\frac{\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2}{T}\le\frac{\mathbb{E}[\mathcal{L}(\boldsymbol{w}^0)]-\mathcal{L}(\boldsymbol{w}^*)}{\eta aT}+\frac{1}{2a}L\eta K^2G^2. \qquad (30)
$$

Proof: Take the expectation of both sides of (29),

$$
\mathbb{E}[\mathcal{L}(\boldsymbol{w}^{t+1})]\le\mathbb{E}[\mathcal{L}(\boldsymbol{w}^t)]-\eta a\mathbb{E}\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2+\frac{1}{2}L\eta^2K^2G^2 \qquad (31)
$$

Adjust the inequality sign of (31),

$$
\eta a\mathbb{E}\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2\le\mathbb{E}[\mathcal{L}(\boldsymbol{w}^t)]-\mathbb{E}[\mathcal{L}(\boldsymbol{w}^{t+1})]+\frac{1}{2}L\eta^2K^2G^2 \qquad (32)
$$

Apply $t=0,1,\dots,T-1$ into (32), and add both sides of these inequalities,

$$
\begin{aligned}
\eta a\sum_{t=0}^{T-1}\mathbb{E}\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2&\le\mathbb{E}[\mathcal{L}(\boldsymbol{w}^0)]-\mathbb{E}[\mathcal{L}(\boldsymbol{w}^T)]\\
&+\frac{1}{2}LT\eta^2K^2G^2\\
&\le\mathbb{E}[\mathcal{L}(\boldsymbol{w}^0)]-\mathcal{L}(\boldsymbol{w}^*)+\frac{1}{2}LT\eta^2K^2G^2 \qquad (33)
\end{aligned}
$$

Let us divide both sides of this inequality by ηaT,

$$
\sum_{t=0}^{T-1}\frac{\|\nabla\mathcal{L}(\boldsymbol{w}^t)\|^2}{T}\le\frac{\mathbb{E}[\mathcal{L}(\boldsymbol{w}^0)]-\mathcal{L}(\boldsymbol{w}^*)}{\eta aT}+\frac{1}{2a}L\eta K^2G^2 \qquad (34)
$$

■

Note that if $\mathbb{E}[\mathcal{L}(\boldsymbol{w}^0)]-\mathcal{L}(\boldsymbol{w}^*)=M$, FedASA converges to a first-order stationary point when $\eta=\sqrt{\frac{2M}{LTK^2G^2}}$.

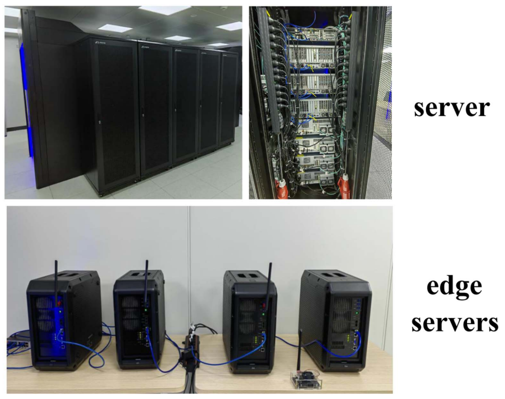


Fig. 8. Network prototype system.

TABLE IV
DESCRIPTIONS OF CWRU DATASET

Dataset	Label	0	1	2	3	4	5	6	7	8	9
CWRU	Fault location	N	IR	IR	IR	BR	BR	BR	OR	OR	OR
	Fault size (mil)	NA	7	14	21	7	14	21	7	14	21

V. EVALUATION

A. Experimental Settings

We carried out our experiments on a prototype system to simulate a real industrial scenario. The prototype system comprises a parameter server and four edge servers. Each edge server has three virtual environments simulating a total of 12 local devices (referred to Fig. 8). The dataset of each device is collected beforehand. The server has an Intel(R) Xeon(R) Silver 4210R CPU and NVIDIA GeForce RTX 3090. The operating system is Ubuntu 20.04. The deep learning models are trained with Pytorch at version 1.12.0.

Dataset: The rolling element bearing dataset from the Case Western Reserve University (CWRU) is collected by sensors deployed at the edge [31]. Sensors are placed on mechanical devices to collect vibration signals in real time. The CWRU dataset includes drive-end (named as location A), fan-end (named as location B), and basic acceleration (named as location C) data which are from three monitoring locations. Four bearing health states are considered, including normal state (N), inner race fault (IR), ball fault (BR), and outer race fault (OR). Each fault type has three fault diameters of 7, 14, and 21 mils. The specific dataset descriptions are shown in Table IV.

Methods for comparison: Several methods have been implemented to evaluate the superiority of FedASA. In particular, we focus on the comparison of the following methods:

- *Local-only* is a baseline method. In this method, each edge device performs model training solely using local dataset.
- *FedAvg* [11] is one of the most famous algorithms in FL, with edge devices sharing the same model architecture.
- *FedProx* [27] utilizes ℓ_2 regularization to constrain the updates of edge devices to mitigate statistical heterogeneity. For FedProx, we set the hyperparameter μ to 0.01.

TABLE V
DEVICE TASKS SETTINGS

Dataset	Task id	Rotating speeds	Load	Location
CWRU	T1-T12	1796 1772 1748 1724	0 1 2 3	A B C

- *FedPer* [30] is a typical G-sharing method that addresses statistical heterogeneity through personalized layers.
- *Ditto* [23] employs ℓ_2 regularization to preserve the association between the global architecture and the local architectures. For Ditto, we set the hyperparameter λ to 0.01.

Tasks & hyperparameters: We evaluate these methods in resource-constrained scenarios with 12 devices on various learning tasks. Table V provides a summary of the configuration of the local dataset. The CWRU dataset encompasses 12 tasks (each device completes one task), each corresponding to four levels of rotating speeds and load, multiplied by three monitoring locations. In the experimental setting, we utilize locations A & B to denote that the dataset is sourced from location A and location B. Similar configurations include locations A & C, locations B & C, and locations A & B & C. As the default configuration, we set the local batch size to 32, the local training epochs to 5, and the global training rounds to 200. Each device is given 1,000 samples, which is split into 80% trainset and 20% testset. The model architecture used by the compared methods is consistent with that of the hyper-model. AI models are updated by the Adam optimizer with a learning rate of 2×10^{-4}. In FedASA, the number of cells for the auxiliary model and hyper-model is $L = M = 4$. The auxiliary data on the parameter server comes from location A, and the accuracy of the auxiliary model is only 78.12%.

B. Performance Evaluation

Firstly, we compared the performance of different baseline methods in cross-location scenarios (referred to Table VI). Fig. 9 illustrates the accuracy of six FL tasks across the four cross-location scenarios. FedASA outperforms other baseline methods, regardless of their monitoring locations, with an average accuracy of greater than 96% and better fairness (the standard deviation is less than 2.5%).

Among baseline methods, the accuracy of local-only remains below 60% with a larger standard deviation. The reason is that the dataset of each device cannot train a high-performing model. In addition, FedAvg and FedProx have demonstrated effective diagnostic performance at locations B & C due to the more extensive similarity between B and C (referred to Fig. 1). The accuracy of FedProx is 86.67% from location B. Yet, this accuracy reduces to 81.13% (referred to Fig. 9)(a) from two locations and further drops to 68.92% (referred to Fig. 9(b)) from three locations. This is because constraint of the update of devices cannot achieve personalized performance. With the increase in monitoring locations, the performance of FedAvg,

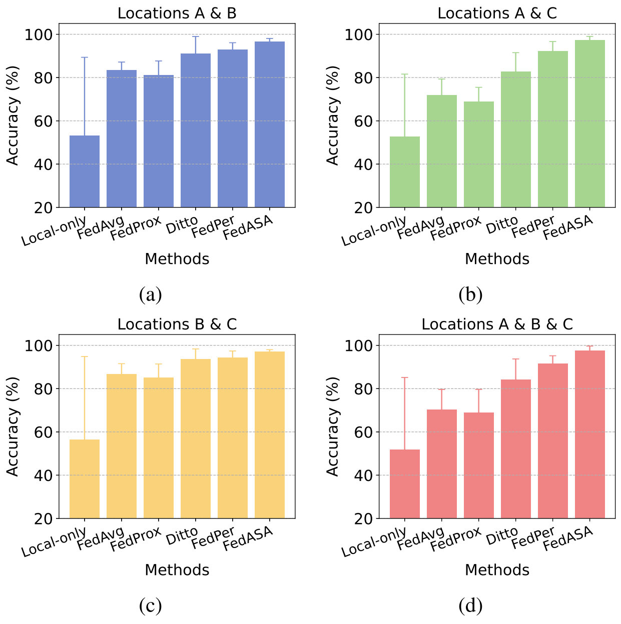


Fig. 9. Accuracy comparison in cross-location scenarios.

FedProx, Ditto, and FedPer declined. The accuracies of Ditto and FedPer drop by 8.75% and 4.83%, respectively, from location B to locations A & B & C. It is demonstrated that all-sharing and G-sharing schemes cannot solve the cross-location issues. As shown in Table VI, the model memory usage of FedASA is limited to only 10.46K to 59.87K, significantly lower than that of all-sharing and G-sharing schemes. FedASA can reduce the bandwidth requirement by 81.49% compared to the all-sharing scheme.

Fig. 10 shows the performance comparison of different methods in three monitoring locations. It can be observed that FedASA outperforms other methods in both accuracy and convergence speed. FedASA achieves an average test accuracy of 97.44% and converges with only 80 rounds of global training. Local-only achieves an accuracy of 51.78% with slow convergence (referred to Fig. 10(a)). Compared with local-only, the accuracies of FedAvg and FedProx are 70.22% and 68.92%, respectively, proving that the FL schemes can improve diagnostic performance. However, their performance is still affected by the cross-location issues. Ditto achieves 84.17% test accuracy, and FedPer achieves 91.42% test accuracy, but they require 200 global training rounds. For them, the cross-location issues not only reduce the diagnostic accuracy but also slow down the model convergence.

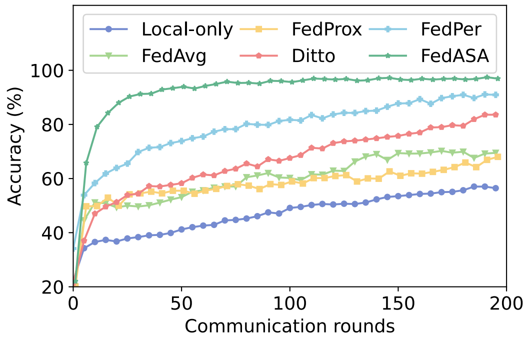
C. Fixed Shared Architectures

Inspired by the G-sharing scheme, Table VI and Fig. 11 show the accuracy comparison of different fixed shared architectures. α_0 to α_4 represent the number of cells in the shared architecture from 0 to 4. Table VI shows that the fixed shared architecture is a good choice for one monitoring location since the accuracies are similar from α_0 to α_4, with an accuracy greater than 93%.

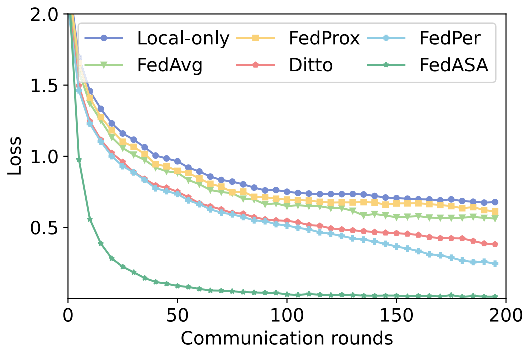
TABLE VI
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE CWRU DATASET

Methods	Model[1]	CWRU Dataset (mean±std %)							Params (K)[2]
		A	B	C	A & B	A & C	B & C	A & B & C	
Local-only	-	51.17±33.85	55.50±38.55	49.92±32.97	53.08±36.29	52.79±28.79	56.50±38.34	51.78±33.38	323.43
FedAvg [11]	all	82.58±1.32	83.25±3.59	86.75±3.44	83.38±3.77	71.88±7.43	86.58±4.95	70.22±9.38	323.43
FedProx [27]	all	81.17±2.27	86.67±2.53	84.08±4.56	81.13±6.56	68.79±6.70	85.08±6.34	68.92±10.73	323.43
Ditto [23]	all	86.33±3.85	92.92±4.15	85.42±7.66	91.00±8.00	82.75±8.76	93.58±4.79	84.17±9.57	323.43
FedPer [30]	G	94.92±1.64	96.25±2.17	96.83±1.89	93.00±3.10	92.29±4.34	94.33±3.08	91.42±3.77	322.14
FedASA	ada	**96.08±0.79**	**96.75±2.19**	**98.00±1.20**	**96.50±1.59**	**97.21±1.83**	**97.00±1.00**	**97.44±2.20**	[10.46, 59.87]
α_0	G	**95.92±1.28**	97.33±0.82	98.17±0.99	95.13±4.00	95.58±2.52	96.83±1.50	96.14±2.64	10.46
α_1	G	95.92±1.53	**98.00±0.53**	98.17±0.69	**96.33±1.36**	96.04±1.38	97.08±1.48	**96.61±2.01**	22.82
α_2	G	94.92±2.10	96.92±2.38	**98.33±0.78**	95.71±1.46	**96.58±2.34**	**97.25±1.83**	96.47±1.63	35.17
α_3	G	93.42±2.06	97.25±1.69	98.50±1.38	93.21±3.59	93.88±2.34	96.83±1.99	95.89±2.47	47.52
α_4	G	95.33±1.87	96.92±1.09	97.00±1.35	90.88±4.48	93.50±2.74	95.17±2.81	93.97±2.86	59.87
ℓ_0	ada	95.50±1.80	**97.92±0.95**	95.42±1.65	95.46±2.04	96.54±2.44	96.75±2.38	96.39±2.56	[10.46, 59.87]
ℓ_1	ada	95.00±2.09	96.83±1.67	98.25±0.76	96.21±2.45	96.21±1.91	**97.42±1.40**	97.00±2.29	[10.46, 59.87]
ℓ_2	ada	**96.08±0.79**	96.75±2.19	98.00±1.20	**96.50±1.59**	**97.21±1.83**	97.00±1.00	**97.44±2.20**	[10.46, 59.87]
ℓ_3	ada	94.08±1.75	98.00±2.35	97.92±1.09	95.79±2.92	95.92±2.96	96.25±1.75	96.31±2.30	[10.46, 59.87]
ℓ_4	ada	95.50±2.46	97.08±2.10	**98.50±1.17**	95.92±2.65	96.37±2.50	96.33±1.96	97.33±1.52	[10.46, 59.87]

1. In the sharing scheme for the models participating in the aggregation, "all" denotes the all-sharing scheme, "G" denotes the G-sharing scheme, and "ada" represents the adaptive-sharing scheme.
2. Memory for the shared architecture.

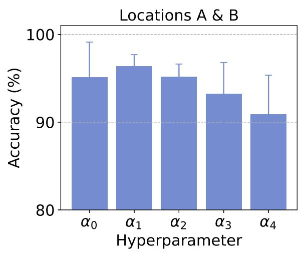


(a) Communication rounds vs. Accuracy (%)

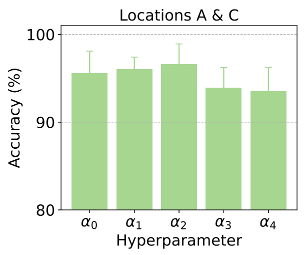


(b) Communication rounds vs. Loss

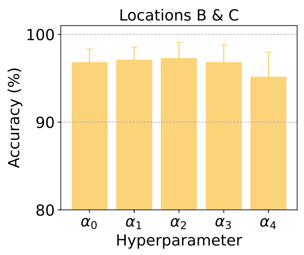
Fig. 10. Performance comparison in three monitoring locations.



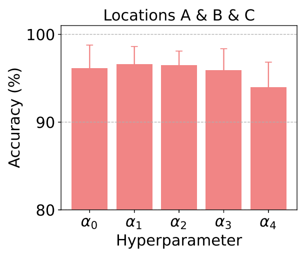
(a)



(b)



(c)



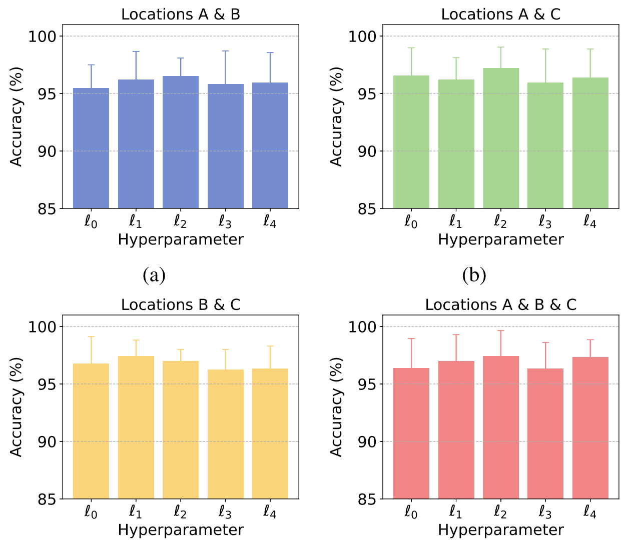
(d)

Fig. 11. Accuracy comparison of different shared architectures.

Fig. 11(a) shows that at locations A & B, the optimal architecture, α_1, achieved an accuracy of 96.33%, while the worst architecture, α_4, is 90.88%. The difference between the two is 5.45%. At locations A & C, this discrepancy is 3.08% (as depicted in Fig. 11(b)), at locations B & C, it is 2.08% (as shown in Fig. 11(c)), and at locations A & B & C, it is 2.64% (in Fig. 11(c)). We have observed that the optimal architecture varies across different scenarios, with discrepancies reaching up to 5.45%. The reason is that each heterogeneous environment requires a customized sharing architecture. Therefore, in cross-position scenarios, fixed-sharing schemes require additional training to search for the optimal number of cells.

D. Hyperparameter Analysis

Table VI and Fig. 12 show the performance comparison of different architectures of the auxiliary model. ℓ_0 to ℓ_4 represent the number of cells in the auxiliary model from 0 to 4. At locations A & B, locations A & C, and locations A & B & C, the optimal hyperparameter is ℓ_2 (referred to Fig. 12(a), (b), and (d)). As shown in Fig. 12(c), at locations B & C, the optimal hyperparameter is ℓ_1. Note that FedASA is not sensitive to the hyperparameter ℓ, and the accuracy is similar among different values. The largest discrepancy is observed in locations A & C, amounting to 1.29%. And the smallest variation



(a) (b) (c) (d)

Fig. 12. Accuracy comparison for different hyperparameters.

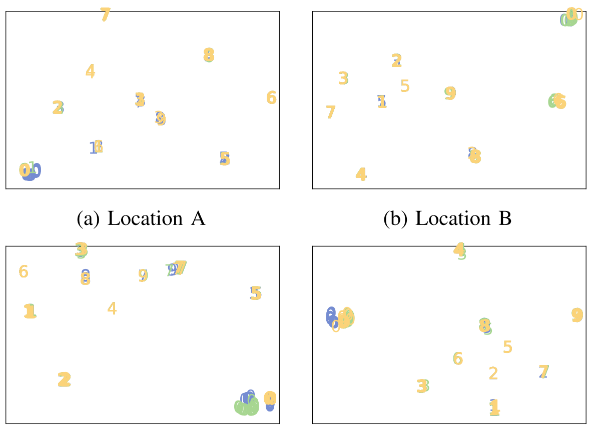

(a) Location A (b) Location B (c) Location C (d) Locations A & B & C

Fig. 13. Feature visualization by the t-SNE method.

is found in locations A & B, with only 1.04%. The reason is that the shared architecture of FedASA can adaptively learn global shared features, reducing differences among devices.

E. Visualization Display

In this subsection, the global features extracted from the shared architectures of different devices are visualized. In particular, the t-SNE method [42] is used to visualize different features. Each label is assigned a specific numerical reference as shown in Table V. Models with good classification performance exhibit characteristics of small intro-group variance and large inter-group variance, meaning that the same numbers are closer together while different numbers are placed farther apart. At locations A, B, or C, different digital colors are used to indicate various working conditions. At locations A & B & C, different colors represent separate monitoring locations. As shown in Fig. 13, FedASA demonstrates exceptional classification performance, with the visualization results showing clear demarcations of the ten categories. We have observed that FedASA exhibits a relatively more considerable intro-group variance when processing samples of healthy status (labeled as 0). This is primarily due to the impact of cross-domain heterogeneity on the healthy samples. Although this effect is less significant compared to cross-location, a certain degree of heterogeneity exists among the samples. As illustrated in Fig. 13(a), FedASA achieves superior classification performance at location A compared to others, which can be attributed to the fact that the auxiliary dataset originates from location A.

VI. Related Work

Federated learning in IoT: With the rise of network bandwidth burden, deploying centralized learning frameworks in IoT scenarios faces several challenges. FL is emerging as a solution that enables joint model training without sharing local data. Mothukuri et al. [43] proposed an FL-based anomaly detection approach to recognize intrusion in IoT networks by decentralized on-device data. Cui et al. [44] introduced a blockchain-empowered decentralized and asynchronous FL framework for anomaly detection in IoT systems. Wang et al. [45] proposed a regional federated learning framework to measure the reliability of vehicles. Cheng et al. [13] proposed a prototype-guided FL framework for human activity recognition. Yao et al. [14] proposed a federated transfer learning framework to address the low data regimes in machine fault diagnosis. Zhang et al. [16] proposed dynamic validation and self-supervision to improve the performance of federated fault diagnosis. However, FL deployment on edge devices faces challenges such as communication constrains, heterogeneous devices, *etc*.

Federated learning with communication constrains: FL is widely deployed on mobile and IoT devices to achieve intelligence. The limitations of communication on these edge devices are often experienced as a bottleneck. Existing methods for saving communication resources include two categories: training multiple local iterations before global aggregation and model compression [20], [46]. The first category reduces the communication overhead by allowing devices to perform multiple epochs of local updates instead of communicating frequently [5], [8], [47], [48]. The second category is to reduce the communication overhead of each round by model compression, so as to reduce the total communication overhead [10], [49], [50]. Different from them, FedASA aims to adaptively determine the shared architecture for uploading according to the heterogeneity of devices, reducing communication overhead by avoiding unnecessary uploading of model architectures.

Federated learning with heterogenous devices: Considering heterogeneous edge devices, FL with heterogeneous devices has been extensively studied in recent studies [51], [52]. Heterogeneity across devices includes resource and statistical heterogeneity. Resource heterogeneity is a straggler issue caused by different computing resources of devices, and several asynchronous strategies have been proposed to address the straggler issue [51], [53], [54], [55]. To address statistical heterogeneity, existing works are divided into data-based methods and model-based methods. Data-based methods focus on constructing independent and identically distributed datasets among devices to reduce the heterogeneity among local datasets [24], [25], [26]. Model-based methods aim to improve the performance of global or personalized models [27], [28], [52], [56], [57]. Different from them, FedASA aims to address the challenges (e.g. statistical heterogeneity, resource constraint, fairness) in a real IoT scenario.

VII. Conclusion

In this paper, we investigated the essential challenges of deploying FL frameworks in real IoT scenarios, i.e., statistical heterogeneity, resource limitations, and fairness. Specifically, the cross-location issues have been demonstrated to introduce more statistical heterogeneity than the cross-domain issues. To address these challenges, we proposed FedASA, a fair and efficient FL method, which can address the challenge of statistical heterogeneity in resource-constrained scenarios by determining the shared architecture adaptively. In FedASA, we first proposed a cell-wised alignment algorithm, which can adaptively determine the shared architecture according to the degree of heterogeneity of each device. We then proposed a cell-based aggregation scheme that aggregates heterogeneous shared architectures. In addition, we provided a theoretical analysis of the federated error bound, which provides a theoretical guarantee for the fairness of FedASA. At the same time, the convergence of FedASA at the first-order stationary point has been proved. Finally, we evaluated the performance of FedASA through extensive simulation and experiments, and experimental results demonstrated that FedASA outperformed the state-of-the-art approaches with better fairness, faster convergence, and less communication requirement.

References

[1] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, “Edge computing in industrial Internet of Things: Architecture, advances and challenges,” *IEEE Commun. Surv. Tut.*, vol. 22, no. 4, pp. 2462–2488, Fourth Quarter, 2020.

[2] T. Zhang et al., “When moving target defense meets attack prediction in digital twins: A convolutional and hierarchical reinforcement learning approach,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3293–3305, Oct. 2023.

[3] T. Zhang, C. Xu, J. Shen, X. Kuang, and L. A. Grieco, “How to disturb network reconnaissance: A moving target defense approach based on deep reinforcement learning,” *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5735–5748, 2023.

[4] J. Yang, C. Wang, B. Jiang, H. Song, and Q. Meng, “Visual perception enabled industry intelligence: State of the art, challenges and prospects,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2204–2219, Mar., 2021.

[5] Z. Zhong et al., “P-FedAvg: Parallelizing federated learning with theoretical guarantees,” in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[6] W. Y. B. Lim et al., “Federated learning in mobile edge networks: A comprehensive survey,” *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter, 2020.

[7] T. Zhang et al., “How to mitigate DDoS intelligently in SD-IoV: A moving target defense approach,” *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 1097–1106, Jan. 2023.

[8] J. Perazzone, S. Wang, M. Ji, and K. S. Chan, “Communication-efficient device scheduling for federated learning using stochastic optimization,” in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1449–1458.

[9] J. Wang, S. Guo, X. Xie, and H. Qi, “Protect privacy from gradient leakage attack in federated learning,” in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 580–589.

[10] H. Liu, F. He, and G. Cao, “Communication-efficient federated learning for heterogeneous edge devices based on adaptive gradient quantization,” 2022, *arXiv:2212.08272*.

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

[12] Z. Zhou, F. Wang, J. Yu, J. Ren, Z. Wang, and W. Gong, “Target-oriented semi-supervised domain adaptation for WiFi-based HAR,” in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 420–429.

[13] D. Cheng, L. Zhang, C. Bu, X. Wang, H. Wu, and A. Song, “ProtoHAR: Prototype guided personalized federated learning for human activity recognition,” *IEEE J. Biomed. Health Informat.*, vol. 28, no. 8, pp. 3900–3911, Aug. 2023.

[14] Z. Yao, C. Jin, M. Ragab, K. M. M. Aung, and X. Li, “DiagNet: Machine fault diagnosis using federated transfer learning in low data regimes,” in *Proc. AAAI Conf. Artif. Intell. Workshop*, 2022.

[15] M. Mehta, S. Chen, H. Tang, and C. Shao, “A federated learning approach to mixed fault diagnosis in rotating machinery,” *J. Manuf. Syst.*, vol. 68, pp. 687–694, 2023.

[16] W. Zhang, X. Li, H. Ma, Z. Luo, and X. Li, “Federated learning for machinery fault diagnosis with dynamic validation and self-supervision,” *Knowl.-Based Syst.*, vol. 213, 2021, Art. no. 106679.

[17] C. Li, S. Li, A. Zhang, Q. He, Z. Liao, and J. Hu, “Meta-learning for few-shot bearing fault diagnosis under complex working conditions,” *Neurocomputing*, vol. 439, pp. 197–211, 2021.

[18] J. Um, J. min Park, S. yeon Park, and G. Yilmaz, “Low-cost mobile augmented reality service for building information modeling,” *Automat. Construction*, vol. 146, 2023, Art. no. 104662.

[19] D. Deng et al., “DecFFD: A personalized federated learning framework for cross-location fault diagnosis,” *IEEE Trans. Ind. Informat.*, vol. 20, no. 5, pp. 7082–7091, May 2024.

[20] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, “Adaptive control of local updating and model compression for efficient federated learning,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5675–5689, Oct. 2023.

[21] A. Vaswani et al., “Attention is all you need,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 998–6008.

[22] T. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang, “Fairness without demographics in repeated loss minimization,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1929–1938.

[23] 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.

[24] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” 2018, *arXiv: 1806.00582*.

[25] 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.

[26] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, “Fine-tuning global model via data-free knowledge distillation for non-iid federated learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10174–10183.

[27] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” in *Proc. MLSys Conf.*, 2020, pp. 429–450.

[28] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, “Scaffold: Stochastic controlled averaging for federated learning,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[29] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, “Personalized federated learning with first order model optimization,” 2021, *arXiv: 2012.08565*.

[30] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, “Federated learning with personalization layers,” 2019, *arXiv: 1912.00818*.

[31] W. A. Smith and R. B. Randall, “Rolling element bearing diagnostics using the case western reserve university data: A benchmark study,” *Mech. Syst. Signal Process.*, vol. 64, pp. 100–131, 2015.

[32] J. Zhang et al., "Fedala: Adaptive local aggregation for personalized federated learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 11237–11244.
[33] J. Xu, X. Tong, and S.-L. Huang, "Personalized federated learning with feature alignment and classifier collaboration," 2023, *arXiv:2306.11867*.
[34] J. R. Regatti, S. Lu, A. Gupta, and N. Shroff, "Conditional moment alignment for improved generalization in federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop*, 2022, pp. 1–13.
[35] P. P. Liang et al., "Think locally, act globally: Federated learning with local and global representations," 2020, *arXiv: 2001.01523*.
[36] X. Ma, J. Zhang, S. Guo, and W. Xu, "Layer-wised model aggregation for personalized federated learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10092–10101.
[37] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–13.
[38] Z. Du, J. Li, H. Su, L. Zhu, and K. Lu, "Cross-domain gradient discrepancy minimization for unsupervised domain adaptation," in *Proc. Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3937–3946.
[39] Y. Zhang, T. Liu, M. Long, and M. Jordan, "Bridging theory and algorithm for domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7404–7413.
[40] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, pp. 151–175, 2010.
[41] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," 2019, *arXiv: 1911.02054*.
[42] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.
[43] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for iot security attacks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2545–2554, Feb. 2022.
[44] L. Cui et al., "Security and privacy-enhanced federated learning for anomaly detection in IoT infrastructures," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3492–3500, May 2022.
[45] S. Wang, F. Liu, and H. Xia, "Content-based vehicle selection and resource allocation for federated learning in IoV," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, 2021, pp. 1–7.
[46] H. Liu, F. He, and G. Cao, "Communication-efficient federated learning for heterogeneous edge devices based on adaptive gradient quantization," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
[47] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Local SGD with periodic averaging: Tighter analysis and adaptive synchronization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–13.
[48] X. Zhao, A. An, J. Liu, and B. X. Chen, "Dynamic stale synchronous parallel distributed training for deep learning," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1507–1517.
[49] E. Ozfatura, K. Ozfatura, and D. Gündüz, "Time-correlated sparsification for communication-efficient federated learning," in *Proc. IEEE Int. Symp. Inf. Theory*, 2021, pp. 461–466.
[50] Y. Mao et al., "Communication-efficient federated learning with adaptive quantization," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–26, 2022.
[51] Z. Wang et al., "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, Sep. 2022.
[52] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 21394–21405.
[53] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.
[54] J. Liu et al., "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 674–690, Feb. 2023.
[55] C.-H. Hu, Z. Chen, and E. G. Larsson, "Scheduling and aggregation design for asynchronous federated learning over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 874–886, Apr. 2023.
[56] M. Asad, A. Moustafa, and T. Ito, "FedOpt: Towards communication efficiency and privacy preservation in federated learning," *Appl. Sci.*, vol. 10, no. 8, 2020, Art. no. 2864.
[57] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," 2020, *arXiv: 2002.07948*.

Dongshang Deng received the BS degree from the Nanjing Agricultural University, Nanjing, China, in 2019. He is currently working toward a master's degree with the Anhui University of Technology, Maanshan, China. His research interests include Internet of Things and federated learning.

Xuangou Wu (Member, IEEE) received the PhD degree in computer software and theory from the University of Science and Technology of China, Hefei, China, in 2013. From 2018 to 2019, he was a visiting scholar with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He is currently a full professor with the School of Computer Science and Technology, Anhui University of Technology, Maanshan, China. His research interests include Internet of Things, network security, and privacy protection.

Tao Zhang (Member, IEEE) received the BS degree in Internet of Things engineering from the Beijing University of Posts and Telecommunications (BUPT) and the Queen Mary University of London in 2018, and the PhD degree in computer science and technology from BUPT in 2023. He is currently an assistant professor (lecturer) with the School of Cyberspace Science and Technology, Beijing Jiaotong University. His publications include ESI highly cited paper and well-archived international journals and proceedings, such as *IEEE Communications Surveys and Tutorials*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Industrial Informatics*, etc. His research interests include network security, moving target defense, and federated learning. He has served as the guest editor for Electronics and Chinese Journal of Network and Information Security, and the TPC chair and a PC member for some international conferences and workshops. He was a recipient of the Best Paper Award from NaNA 2018 and IWCMC 2021, and a recipient of Outstanding Paper Award from IEEE iThings 2023. His Ph.D. thesis was awarded the Outstanding Doctoral Dissertation by BUPT in 2023.

Xiangyun Tang received the BEng degree in computer science from the Minzu University of China, Beijing, China in 2016, and the PhD degree in cyberspace security from the Beijing Institute of Technology, Beijing, China, in 2022. She is an associate professor with the School of Information Engineering, Minzu University of China. She has served as the guest editor for multiple Journals, and the TPC chair and the PC member for international conferences and workshops. She was a recipient of the Best Paper Award from IEEE ICBCTIS 2023, and a recipient of Outstanding Paper Award from IEEE iThings 2023. Her research interests include secure multi-party computation and machine learning security.

Hongyang Du received the BEng degree from the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, in 2021, and the PhD degree from the Interdisciplinary Graduate Program from the College of Computing and Data Science, Energy Research Institute @ NTU, Nanyang Technological University, Singapore, in 2024. He is an assistant professor with the Department of Electrical and Electronic Engineering, University of Hong Kong. He serves as the editor-in-chief assistant of *IEEE Communications Surveys & Tutorials* (2022-2024), and the guest editor for IEEE VTM. He is the recipient of the IEEE Daniel E. Noble Fellowship Award from the IEEE Vehicular Technology Society in 2022, the IEEE Signal Processing Society Scholarship from the IEEE Signal Processing Society in 2023, the Chinese Government Award for Outstanding Students Abroad in 2023, the Singapore Data Science Consortium (SDSC) Dissertation Research Fellowshipin 2023, and NTU Graduate College's Research Excellence Award in 2024. As the team leader, he won the Honorary Mention award in the ComSoc Student Competition from IEEE Communications Society in 2023, and the First and Second Prizes in the 2024 ComSoc Social Network Technical Committee (SNTC) Student Competition. He was recognizedas an exemplary reviewer of the *IEEE Transactions on Communications and IEEE Communications Letters* in 2021. His research interests include edge intelligence, generative AI, semantic communications, and network management.

Jiawen Kang (Senior Member, IEEE) received the PhD degree from the Guangdong University of Technology, China, in 2018. He was a postdoc with Nanyang Technological University, Singapore, from 2018 to 2021. He currently is a professor with the Guangdong University of Technology, China. His research interests mainly include focus on blockchain, security, and privacy protection in wireless communications, and networking.

Jiqiang Liu (Senior Member, IEEE) received the PhD degree from Beijing Normal University in 1999. He is currently a full professor and the dean with the School of Cyberspace Science and Technology, Beijing Jiaotong University. He has authored or coauthored more than 200 publications. In recent years, he has been mainly engaged in research on trusted computing, privacy protection, and cloud computing security.

Dusit Niyato (Fellow, IEEE) received the BEng degree from the King Mongkuts Institute of Technology Ladkrabang (KMITL), Thailand, and the PhD degree in electrical and computer engineering from the University of Manitoba, Canada. He is a professor with the College of Computing and Data Science, Nanyang Technological University, Singapore. His research interests include the areas of mobile generative AI, edge intelligence, decentralized machine learning, and incentive mechanism design.