FedCache: A Knowledge Cache-Driven Federated Learning Architecture for Personalized Edge Intelligence

Zhiyuan Wu, *Member, IEEE*, Sheng Sun, Yuwei Wang, *Member, IEEE*, Min Liu, *Senior Member, IEEE*, Ke Xu, *Fellow, IEEE*, Wen Wang, Xuefeng Jiang, Bo Gao, *Member, IEEE*, and Jinda Lu

***Abstract*—Edge Intelligence (EI) allows Artificial Intelligence (AI) applications to run at the edge, where data analysis and decision-making can be performed in real-time and close to data sources. To protect data privacy and unify data silos distributed among end devices in EI, Federated Learning (FL) is proposed for collaborative training of shared AI models across multiple devices without compromising data privacy. However, the prevailing FL approaches cannot guarantee model generalization and adaptation on heterogeneous clients. Recently, Personalized Federated Learning (PFL) has drawn growing awareness in EI, as it enables a productive balance between local-specific training requirements inherent in devices and global-generalized optimization objectives for satisfactory performance. However, most existing PFL methods are based on the Parameters Interaction-based Architecture (PIA) represented by FedAvg, which suffers from unaffordable communication burdens due to large-scale parameters transmission between devices and the edge server. In contrast, Logits Interaction-based Architecture (LIA) allows to update model parameters with logits transfer and gains the advantages of communication lightweight and heterogeneous on-device model allowance compared to PIA. Nevertheless, previous LIA methods attempt to achieve satisfactory performance either relying on unrealistic public datasets or increasing communication overhead for additional information transmission other than logits. To tackle this dilemma, we propose a knowledge cache-driven PFL architecture, named FedCache, which reserves a knowledge cache on the server for fetching personalized knowledge from the samples with similar hashes to each given on-device sample. During the training phase, ensemble distillation is applied to on-device models for constructive optimization with personalized knowledge transferred from the server-side knowledge cache. Empirical experiments on four datasets demonstrate that FedCache achieves comparable performance with state-of-art PFL approaches, with more than two orders of magnitude improvements in communication efficiency.**

***Index Terms*—Distributed architecture, edge computing, personalized federated learning, knowledge distillation, communication efficiency.**

Manuscript received 31 August 2023; revised 5 November 2023; accepted 31 January 2024. Date of publication 5 February 2024; date of current version 3 September 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2900102, in part by the National Natural Science Foundation of China under Gant 62072436, in part by the Innovation Capability Support Program of Shaanxi under Grant 2023-CX-TD-08, in part by Shaanxi Qinchuangyuan "scientists+engineers" team under Grant 2023KXJ-040, and in part by the Innovation Funding of ICT, CAS under Grant E261080. Recommended for acceptance by X. Yuan. *(Corresponding author: Yuwei Wang.)*

Zhiyuan Wu, Wen Wang, and Xuefeng Jiang are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China, and also with the University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: wuzhiyuan22s@ict.ac.cn; wangwen22s@ict.ac.cn; jiangxuefeng21b@ict.ac.cn).

Sheng Sun and Yuwei Wang are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China (e-mail: sunsheng@ict.ac.cn; ywwang@ict.ac.cn).

Min Liu is with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China, and also with Zhongguancun Laboratory, Beijing 100086, China (e-mail: liumin@ict.ac.cn).

Ke Xu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100190, China, and also with Zhongguancun Laboratory, Beijing 100086, China (e-mail: xuke@tsinghua.edu.cn).

Bo Gao is with the School of Computer and Information Technology, Engineering Research Center of Network Management Technology for High-Speed Railway of Ministry of Education, Beijing Jiaotong University, Beijing 100082, China (e-mail: bogao@bjtu.edu.cn).

Jinda Lu is with the School of Information Science and Technology, University of Science and Technology of China, Hefei 101127, China (e-mail: lujd@mail.ustc.edu.cn).

Our code and DEMO are available at https://github.com/wuzhiyuan2000/FedCache.

Digital Object Identifier 10.1109/TMC.2024.3361876

I. Introduction

EDGE Intelligence (EI) is an emerging technology for the marriage of edge computing and Artificial Intelligence (AI), enabling real-time data analysis and decision-making close to data sources instead of relying solely on the cloud [1]. With the proliferation of mobile devices and the unprecedented amount of data generated by ubiquitous devices, EI is playing an increasingly important role in many areas such as unmanned vehicles [2], smart homes [3], recommender systems [4], etc. However, conventional centralized EI paradigms require uploading raw data for training pervasive AI models, raising privacy concerns about sensitive data leakage.

Federated Learning (FL) is a privacy-preserving distributed learning paradigm that enables multiple data owners to collaboratively train AI models without sharing owners' private data. Due to the benefits of data localization and privacy protection, FL has shown great potential in various EI applications, such as healthcare [5], smart transportation [6], industrial manufacturing [7], etc. Unfortunately, the prevailing FL approaches [8], [9] require all participating devices (named clients) to share a uniform model, which is extremely difficult to deploy and generalize to all devices because of the inherent characteristics of device variation in terms of data heterogeneity, resources limitation, task differentiation, etc [10], [11]. Recent studies pay much attention to Personalized Federated Learning (PFL) [11]

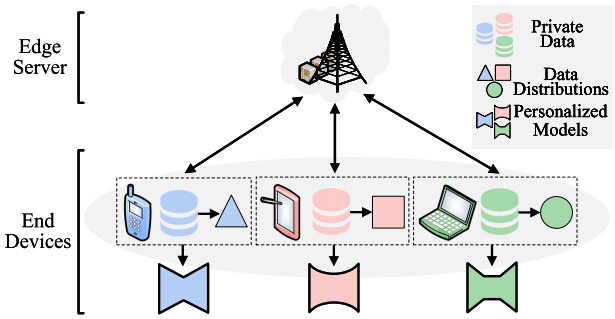

Fig. 1. Schematic diagram of personalized federated learning for edge intelligence.

for addressing the differential training challenges in EI via building personalized models for individual devices, as shown in Fig. 1. However, most PFL approaches [12], [13], [14] adopt the Parameters Interaction-based Architecture (PIA) represented by FedAvg [8], which requires homogeneity of on-device model architectures and imposes tremendous communication burden caused by large-scale parameters exchange between clients and the server for bandwidth-limited devices [15], [16].

Furthermore, by applying knowledge distillation technology [17], [18], [19] to PFL, a series of communication-lightweight and heterogeneous model-allowable PFL architectures with logits (usually called knowledge) exchange instead of interacting model parameters are put forward. These architectures, which we call Logits Interaction-based Architecture (LIA), bring the benefits of saving orders of magnitude of communication overhead and training models with heterogeneous architectures. Related literatures [16], [20], [21], [22], [23] fall into two types of architectures based on the granularity of the interacted logits during training: Class-grained Logits Interaction-based Architecture (CLIA) and Sample-grained Logits Interaction-based Architecture (SLIA). Thereinto, SLIA is drawn more attention since it allows for fine-grained interaction of logits for performance guarantee. However, existing methods based on SLIA endeavor to achieve satisfactory performance either relying on additional client-side training on unrealistic public datasets [20], [21], or requiring the transfer of embedded features with non-negligible sizes in addition to logits [22], [24]. They appear to be unfriendly to devices by reason of the induction of intensive computation, tremendous communication, or public datasets reliance, making them unsuitable for practical applications in EI [25].

In this paper, we develop a novel device-friendly PFL architecture that is suitable for EI, named knowledge cache-driven FL architecture (FedCache). FedCache is a novel client-server interaction paradigm, which maintains a knowledge cache on the server to store the latest knowledge associated with each private sample, and applies a customized knowledge cache-driven personalized distillation technique for on-device model training. During the initialization process, all private samples on clients are encoded into hashes via a deep pre-trained neural network, so as to discern the relational degree among samples in a privacy-preserving manner. During the training process, each on-device model is optimized via personalized knowledge distillation over the ensemble of relevant knowledge whose corresponding hashes are the R-nearest neighbors of the hash of the given sample to be optimized on, which is fetched from the server-side knowledge cache. To our best knowledge, FedCache is *the first Sample-grained Logits Interaction-based Architecture (SLIA) dispensed with features transmission and public datasets*, ensuring the satisfactory performance of on-device models while meeting the practical limitations of EI.

In general, we summarize the contributions of our proposed FedCache as follows:

- *Device Friendliness:* FedCache is a device-friendly architecture that enables only small-scale ensemble logits to be transferred between clients and the server during training without needing public datasets. Meanwhile, FedCache supports collaborative training on devices with heterogeneous models.
- *Scalability:* FedCache is a highly scalable architecture for large-scale devices since it eliminates the need to keep a cumbersome global model on the server and also enables asynchronous training, effectively reducing the server-side computation and client-server synchronization consumption.
- *Effectiveness:* FedCache is compared with state-of-art PFL methods with various architectures on four common datasets. Results confirm that FedCache achieves performance comparable to benchmark algorithms while improving communication efficiency by two orders of magnitude.

II. Related Work

A. Federated Learning for Personalized Edge Intelligence

As a uniform shared model cannot accommodate multiple clients with diverse tasks and capabilities [10], [26], personalization techniques in FL are needed to adapt clients-side individualized requirements. Specifically, federated multi-task learning in edge computing [13], [22] allows clients to train personalized neural networks to accommodate the differentiated data distribution in their respective tasks. [14] retains historical personalized models on devices, allowing current models to distill knowledge from previous models for personalized EI. [27] leverages a source-free unsupervised domain adaptation approach to adapt large source-domain models to target data on devices, while adopting lite residual hypothesis transfer to save the storage overhead during the adaptation process. In addition, [28] considers the personalization of accuracy targets on clients and uses adaptive learning rates to allow clients that reach the target to exit in advance for saving resources.

Unlike prior works, we focus on the architecture design of PFL to enable efficient communication as well as asynchronous training, while also supporting heterogeneous on-device models without requiring public datasets, aiming to bridge the gap between PFL and practical applications in EI.

B. Knowledge Distillation in Federated Edge Learning

As noted in [25], knowledge distillation has emerged as an important technique for addressing challenges in federated edge

learning. [23], [29] leverage knowledge distillation as a communication protocol for exchanging model representations among devices and the edge server, enabling communication-efficient training over heterogeneous models. [30], [31] transfer knowledge from edge models to differentiated on-device models in FL under device heterogeneity. [14], [22] conduct distillation-based personalized FL optimization over on-device models to support personalized EI.

However, the above approaches cannot achieve satisfactory performance in a device-friendly manner, since they either need features/model parameters transfer or rely on public data during the training process. In contrast, our FedCache architecture solves all the above-mentioned problems and enables practical use in personalized EI.

III. Preliminary and Motivation

A. Background and Notations

We investigate PFL for EI, where distributed devices (named clients) collaboratively train C-class classification models coordinated by an edge server (named the server) while keeping private data on devices. We assume that K clients participate in PFL, and each client $k \in \{1, 2, \ldots, K\}$ occupies a private dataset $\mathcal{D}^k := \bigcup_{i=1}^{N^k} \{(X_i^k, y_i^k)\}$, where N^k is the number of samples in $\mathcal{D}^k$, and X_i^k, y_i^k are the i-th data and label in $\mathcal{D}^k$, respectively. Each device k owns a personalized model $M^k := (W^k, f^k)$ with possible different model parameters or architectures, where W^k is the model parameters of M^k and $f^k(\cdot)$ is the non-linear mapping determined by M^k. The goal of each device is to improve the User model Accuracy (UA) [13] of its personalized model on its private data as much as possible. The optimization objective of the PFL system is to maximize the Maximum Average UA (MAUA) of all clients, that is to achieve generally satisfactory performance on each client. The main notations and descriptions can be referred to Table I.

TABLE I
Main Notations With Descriptions

Notation	Description
C	The number of classes
K	The number of clients
$\mathcal{D}^k$	The local dataset of client k
X_i^k	The i-th sample of $\mathcal{D}^k$
y_i^k	The label of X_i^k
N^k	The number of samples in $\mathcal{D}^k$
M^k	The personalized model on client k
W^k	The model parameters of M^k
f^k	The non-linear mapping determined by M^k
f^h	The hash mapping determined by a pre-trained deep neural network
L_{CE}	The cross-entropy loss
KL	The Kullback-Leibler divergence loss
σ_0	The softmax mapping
LI	The label-to-index pairs
IK	The index-to-knowledge pairs
IH	The index-to-hash pairs
IR	The index relations pairs
KC	The knowledge cache
R	The number of related samples in KC
h_i^k	The hash value of X_i^k
z_i^k	The knowledge of X_i^k
$(zr_i^k)_s$	The s-th knowledge fetched for the given sample index (k, i)
$\overline{zr}_i^k$	The ensembled fetched knowledge for the given sample index (k, i)
J^k	The optimization objective of M^k

B. Practical Limitations in Edge Intelligence

We summarize the main practical limitations that PFL architectures need to overcome when deploying in EI:

- *Device Heterogeneity:* Considering the different hardware configurations of end devices such as central processing units, memory resources, and energy status, personalized models need to be adopted among devices to fit their specific characteristics [32], [33].
- *Communication Efficiency:* Due to the limited wireless network bandwidth between edge servers and end devices, they are not capable of large-scale communication [16], [32].
- *Data Privacy:* Devices are reluctant to share their local data with edge servers because of privacy concerns or data protection regulations [34], [35]. Hence, it is difficult to obtain information about users' local data.
- *Asynchronous Optimization:* The high synchronization overhead caused by varying computation tasks, capabilities, and communication delays of different devices impedes model update [36], [37].

C. Overview of PFL Architectures

1) PFL Architecture Based on Parameters Interaction: For Parameters Interaction-based Architecture (PIA), each client periodically uploads locally-trained model parameters to the server and updates the local model with the server-downloaded model parameters obtained from aggregating local models. In PFL with PIA, clients tend to upload only part of its model parameters to preserve local adaptation capabilities [13], [14]. Therefore, filtered parameters aggregation weighted by local sample numbers is performed on the server side, that is:

$$W^* = \frac{N^k}{\sum_{l=1}^{K} N^l} \cdot filter(W^k), \tag{1}$$

where $filter(\cdot)$ filters out partial on-device model parameters to be uploaded to the server, and W^* represents the aggregated model parameters on the server.

Although PIA can preserve the personalization capabilities of on-device models by filtering model parameters, transmitting large-scale parameters for aggregation is still too costly

for devices with limited communication resources [16], [32]. Moreover, PIA demands a high degree of homogeneity among on-device model architectures during the aggregation process, which is hard to achieve in edge intelligence scenarios where heterogeneous devices with various hardware-related constraints are prevalent [38], [39].

2) PFL Architecture Based on Logits Interaction: For Logits Interaction-based Architecture (LIA), each client performs distillation-based optimization on the global logits downloaded from the server, without parameters transmission during training [22], [23], [24], [29], [40], [41]. Depending on the granularity of interacted logits, existing PFL architectures can be divided into two categories: class-grained logits interaction and sample-grained logits interaction.

1) Class-grained Logits Interaction-based Architecture (CLIA): For CLIA, the output of each sample X_i^k from client k needs to approach the global average logits calculated by all samples with the same label y_i^k from all other clients except client k [23], that is:

$$\arg\min_{W^k} \sum_{(X_i^k, y_i^k)\in\mathcal{D}^k} \Bigg[L_{CE}(\sigma_0(f^k(X_i^k)), y_i^k) + \gamma \cdot L_{CE}\left(\sigma_0(f^k(X_i^k)), \sigma_0\left(\frac{\sum_{l=1}^{K} F^{l,y_i^k} - F^{k,y_i^k}}{K-1}\right)\right)\Bigg], \tag{2}$$

where $\sigma_0(\cdot)$ is the softmax mapping, γ is the distillation weighting factor, and $L_{CE}(\cdot)$ denotes the cross-entropy loss. F^{l,y_i^l} is the average logits calculated by the samples with the same label y_i^l in client l, i.e.

$$F^{l,y_i^l} = \underset{(X_i^k, y_i^l)\in\mathcal{D}^k \wedge y_i^l = y_i^k}{E} f^k(X_i^k). \tag{3}$$

Although CLIA supports model heterogeneity with lightweight communication, it only enables C types of logits to be learned by each client. As clients learn very little additional server-side information compared to standalone, this PFL design is prone to reaching a performance limit.

2) Sample-grained Logits Interaction-based Architecture (SLIA): For SLIA, the number of logits learned by on-device models are related to the number of samples [22], [24], [29], [40], [41]. Such architecture generally requires inevitable compromises of importing public datasets or increasing communication overhead, and can be classified into two forms.

- *SLIA with Features Exchange (SLIA-FE):* In SLIA-FE, the model parameters of client k are divided into the feature extractor part W_e^k and the predictor part W_p^k, where the prediction mapping of the feature extractor is denoted as $f_e^k(\cdot)$. The server keeps only a large-scale classifier W^S with the corresponding prediction mapping $f^S(\cdot)$. Typically, the model on the server is updated with a linear combination of cross-entropy loss $L_{CE}(\cdot)$ and Kullback-Leibler divergence loss $KL(\cdot)$ depending on clients-side uploaded features and logits [22], [24], [40], which can be expressed as follows:

$$\arg\min_{W^S} \sum_{(X_i^k, y_i^k)\in\mathcal{D}^k} [L_{CE}(\sigma_0(f^S(\underbrace{f_e^k(X_i^k)}_{\text{uploaded features}})), y_i^k) + \lambda \cdot KL(\sigma_0(f^S(\underbrace{f_e^k(X_i^k)}_{\text{uploaded features}}))||\sigma_1(\underbrace{f^k(X_i^k)}_{\text{uploaded logits}}))], \tag{4}$$

where $\sigma_1(\cdot)$ is the transform mapping for local logits, and λ is the distillation weighting factor. Contrastively, client k performs local model parameters update with the server-side downloaded global logits, and optimizes the following loss function:

$$\arg\min_{W^k} \sum_{(X_i^k, y_i^k)\in\mathcal{D}^k} \Bigg[L_{CE}(\sigma_0(f^k(X_i^k)), y_i^k) + \mu \cdot KL \times \Bigg(\sigma_0(f^k(X_i^k))||\sigma_2\Bigg(\underbrace{f^S\Bigg(\underbrace{f_e^k(X_i^k)}_{\text{uploaded features}}\Bigg)}_{\text{downloaded global logits}}\Bigg)\Bigg)\Bigg], \tag{5}$$

where $\sigma_2(\cdot)$ is the transform mapping for global logits, and μ is the distillation weighting factor. Although SLIA-FE allows for heterogeneous on-device models without parameters transmission, participants need to agree on the feature dimensionality. Besides, since the feature dimensionality of high-resolution images and long sequential data is often high, the overhead of features transmission is still significant for devices.

- *SLIA with Public Dataset (SLIA-PD):* For SLIA-PD, client k aims to approach the average logits of all clients on a given sample (X_i^O, y_i^O) in the public dataset $\mathcal{D}^O$ [29], [41], that is:

$$\arg\min_{W^k} \sum_{(X_i^O, y_i^O)\in\mathcal{D}^O} L_{CE}\left(\sigma_0(f^k(X_i^O)), \sigma_0\left(\frac{1}{K}\sum_l \frac{f^l(X_i^O)}{U}\right)\right), \tag{6}$$

where U is a hyper-parameter that controls the distribution of ensembled logits. We claim that SLIA-PD not only further relaxes the constraints on model architectures across clients, but also enables exchanges of only logits with minuscule sizes during training, resulting in significantly lower communication overhead compared to previously mentioned architectures. However, SLIA-PD relies on a public dataset whose distribution should be close to private data on clients [42]. As it is unlikely to collect satisfactory public data without knowing data distribution of clients, this architecture is impractical in reality.

TABLE II
COMPARISON OF FEDCACHE WITH OTHER PFL ARCHITECTURES IN TERMS OF MODEL HETEROGENEITY SUPPORTABILITY, COMMUNICATION EFFICIENCY, DEPENDENCY ON PUBLIC DATA, WHETHER ENABLE ASYNCHRONOUS OPTIMIZATION, AND COMMUNICATION PROTOCOL

Architecture	Model Hetero. Supportability	Comm. Efficiency	No Dependency on Public Data	Asynchronous Optimization	Communication Protocol
PIA	Partial Hetero.	Low	Yes	No	Model Parameters
CLIA	Complete Hetero.	High	Yes	No	Class-grained Logits
SLIA-FE	Complete Hetero. with Features Dim. Agreement	Medium	Yes	Yes	Sample-grained Features and Logits
SLIA-PD	Complete Hetero.	High	No	No	Sample-grained Logits
FedCache	**Complete Hetero.**	**High**	**Yes**	**Yes**	**Sample-grained Logits**

D. Motivation

From the above analysis, we can conclude that existing PFL architectures cannot realize well-satisfied trade-offs among system performance, resource efficiency and without relying on public datasets, even if LIA gains advantages of remarkably reducing communication burden and tolerating heterogeneous models training over frequently-used PIA. Motivated by the analysis above about PFL architectures, we attempt to answer the following question: *how can a personalized federated learning architecture be designed to allow only logits transmission during the training process without the need for a public dataset, meanwhile significantly outperforming class-grained logits interaction-based architecture?* Concisely, our answer is to develop a knowledge cache-driven federated learning architecture with personalized distillation to optimize local models on clients.

To optimize on-device models via knowledge distillation, we propose to keep a knowledge cache on the server, which serves as the source of sample-grained knowledge for personalized distillation without public datasets. Specifically, the server-side knowledge cache keeps track of the latest knowledge of samples and leverages an information retrieval mechanism to search out the most relevant knowledge for each sample from cached knowledge. The searched knowledge from other clients is accompanied by reliable and effectual relevant representations, and is transferred to clients from which the sample originated for constructive distillation-based optimization. On this basis, sample-grained logits interaction can be realized between the server and clients to ensure that on-device models learn sufficient personalized knowledge.

Based on the above insights, FedCache is proposed, whose comparisons with other FL architectures are shown in Table II. Compared to existing architectures, FedCache supports transferring sample-level logits without the assistance of public datasets during training, achieving superior performance compared to CLIA and overcoming the drawbacks of previous SLIA. Besides, FedCache is a device-friendly architecture that enables complete model heterogeneity among clients, unlike other existing approaches either requiring partial model homogeneity or agreeing on the same feature dimension. In addition, FedCache supports asynchronous interaction of logits required for PFL systems with devices of different capabilities, since it does not need to synchronously aggregate logits from different clients unlike previous methods [23], [29], [41].

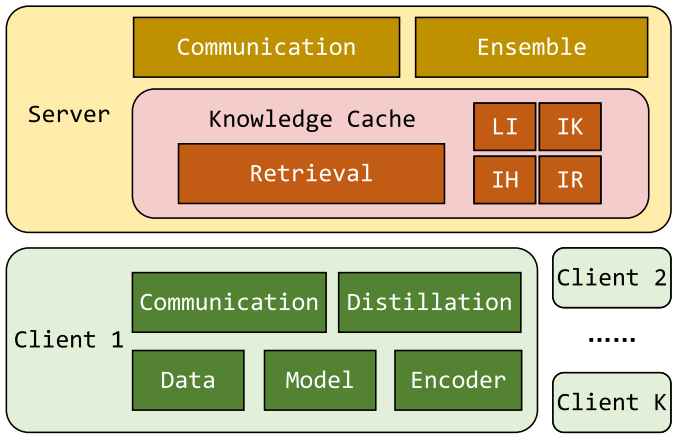

Fig. 2. Functional module diagram of FedCache.

IV. KNOWLEDGE CACHE-DRIVEN PERSONALIZED FEDERATED LEARNING

A. System Design

The functional module diagram of FedCache is displayed in Fig. 2, which consists of a server with three functional modules: (server-client) communication, (knowledge) ensemble, knowledge cache; and K clients with five functional modules: (client-server) communication, (knowledge) distillation, data, model, and (sample) encoder. Specifically, the ensemble module combines the fetched knowledge from the knowledge cache to obtain personalized knowledge to be distilled over clients; the knowledge cache module is our designed self-organizing knowledge storage structure that facilitates fetching each client's relevant knowledge on the server side; the model module extracts knowledge from local data, and conducts model updates under the guidance of the distillation module; in addition, the encoder module encodes private data into hash values to initialize the knowledge cache. The encoder should be efficient, robust, and discriminative, ensuring that the hash values of local samples can be computed quickly and reliably reflect the semantic similarity among samples.

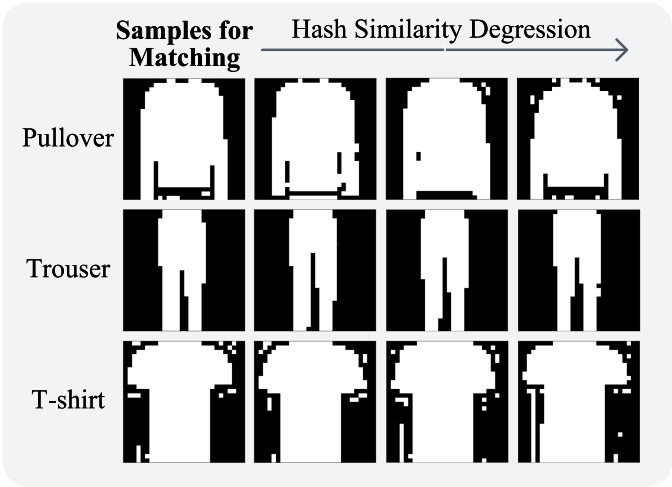

Fig. 3. Sample matching results on FashionMNIST dataset with $R = 3$.

During the initialization phase, the generated hash codes on clients are uploaded to the server in a single pass. Then, HNSW [43] is performed in the server-side knowledge cache, aiming to retrieve R most relevant samples for matching each sample measured by cosine similarity of hash values. Fig. 3 displays the sample matching results on FashionMNIST [44] dataset. As shown, the matched samples are very similar to the original sample, making the knowledge extracted from them beneficial to client-side distillation on the original sample. During the training phase, each logits and index of private samples are uploaded to the server in each communication round. Then, R best-matching knowledge with the highest hash similarity in the knowledge cache for each sample is fetched based on the pre-established similarity relations, followed by knowledge ensemble and then knowledge communication to corresponding clients for local distillation. As the distillation phase only relies on highly relevant knowledge of clients' respective private data, the resulting model is locally adaptable and powerful for personalization tasks. We will introduce the key procedures of FedCache in the subsequent subsections.

B. Knowledge Cache

The knowledge cache on the server is proposed to asynchronously fetch relevant knowledge for an arbitrary local sample with controllable computation complexity, where the corresponding hash values of samples from which relevant knowledge is extracted should be one of the R-nearest neighbors of the hash value of the original sample. Guided by the above design, we preserve multiple pairs in the knowledge cache, including label-to-index pairs (LI), index-to-knowledge pairs (IK), index-to-hash pairs (IH), and index relations pairs (IR), where each pair enables mapping the first element to the second element. On this basis, the knowledge cache is of two main phases: initialization and training.

The initialization process includes the following steps:

- *Pairs initialization:* The uploaded hash value h_i^k corresponding to each sample index (k, i) is stored in IH. In addition, indexes are added to LI according to their corresponding label classes, and the knowledge corresponding to each given index is initialized to zeros in IK, i.e.

$$IH(k,i) \leftarrow h_i^k, \tag{7}$$

$$LI(y_i^k) \leftarrow LI(y_i^k) \cup \{(k,i)\}, \tag{8}$$

$$IK(k,i) \leftarrow (\underbrace{0,\ldots,0}_{Czeros}). \tag{9}$$

  As LI only allows relations to be built within the sample index range of the same label class, it is expected that the number of candidate samples used for matching will be reduced, improving the computation efficiency of the relations establishment in the following step.
- *Build relations:* For each given sample index (k, i), we relate it to R indexes $\{(l_1, j_1), (l_2, j_2), \ldots, (l_R, j_R)\}$ whose hash values have the greatest cosine similarity to the hash value of the given sample among all the candidate hashes, i.e.

$$\begin{aligned} &\underset{(l_1,j_1),(l_2,j_2),\ldots,(l_R,j_R)}{\arg\max} \sum_{m=1}^{R} \cos(IH(k,i), IH(l_m,j_m)), \\ &\text{s.t.} \begin{cases} l_{n_1} \neq l_{n_2} \vee j_{n_1} \neq j_{n_2}, \forall n_1, n_2 \wedge n_1 \neq n_2, \\ (k,i) \in LI(y^*) \wedge (l_m, j_m) \in LI(y^*), \exists y^*, \\ n_1, n_2, m \in \{1,2,\ldots,R\}, \\ y^* \in \{1,2,\ldots,C\}, \end{cases} \end{aligned} \tag{10}$$

  during which HNSW [43] is adopted to achieve the R-nearest neighbors retrieval. Then, the retrieved results related to each sample index are saved in IR for subsequent access, i.e.

$$IR(k,i) \leftarrow \{(l_1, j_1), (l_2, j_2), \ldots, (l_R, j_R)\} \tag{11}$$

During the training process, the following steps should be performed for each given sample index:

- *Knowledge fetching:* The most relevant knowledge can be fetched in the knowledge cache KC based on a provided sample index: for a newly uploaded sample index (k, i), the corresponding knowledge is obtained and returned according to 1) IR which stores relevant sample indexes of (k, i), and 2) IK which transforms relevant indexes to knowledge, that is:

$$KC(h_i^k; k, i) = IK(IR(k, i)). \tag{12}$$

  As knowledge fetching requires only the clients requesting knowledge to be online, clients can asynchronously perform fetched knowledge-based optimization.
- *Knowledge update:* $IK(k, i)$ is updated with the knowledge z_i^k corresponding to the given sample index (k, i), so that the latest knowledge can be fetched on the next access, i.e.

$$IK(k,i) \leftarrow z_i^k. \tag{13}$$

C. Knowledge Cache-Driven Personalized Distillation

We optimize on-device models with personalized federated distillation, where knowledge of samples similar to each client's private data is fetched from the knowledge cache. On this basis,

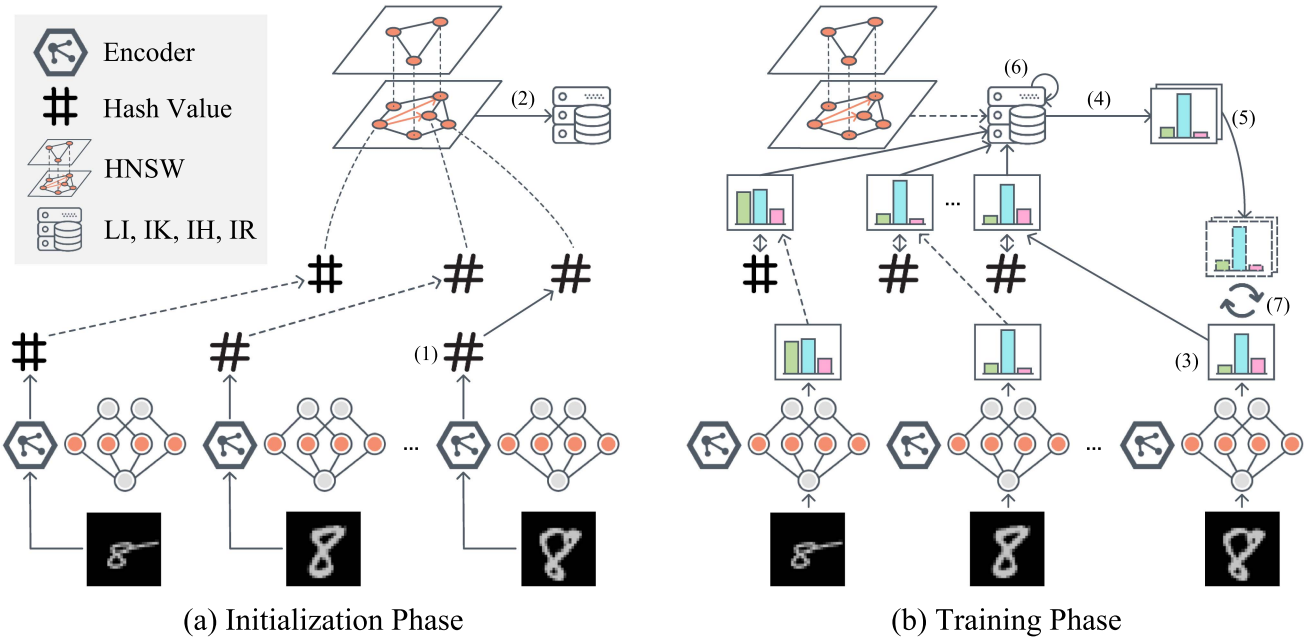

Fig. 4. Overview of executing procedure of FedCache. (1) Hash encoding and uploading. (2) Knowledge cache initialization. (3) Knowledge extraction and uploading. (4) Knowledge fetching. (5) Knowledge ensemble and distributing. (6) Knowledge update. (7) Knowledge acceptance and distillation.

each client performs ensemble distillation on fetched knowledge for constructive optimization of on-device models. Specifically, a pre-trained deep neural network $f^h(\cdot)$ is adopted as an encoder to generate the hash values of samples on clients during initialization, i.e.

$$h_i^k = f^h(X_i^k), \tag{14}$$

and such hash values with corresponding sample indexes and labels are uploaded to the server for initializing the knowledge cache according to (7), (8), (9), (10), and (11).

During training on each given sample (X_i^k, y_i^k), client k first extracts knowledge z_i^k on X_i^k, and then uploads z_i^k with corresponding sample index (k, i) to the server, where:

$$z_i^k = f^k(X_i^k). \tag{15}$$

Then, the R knowledge related to sample index (k, i) is fetched from the knowledge cache KC according to (12), that is,

$$(zr_i^k)_1, (zr_i^k)_2, \ldots, (zr_i^k)_R = KC(h_i^k; k, i), \tag{16}$$

where $(zr_i^k)_s$ is the s-th knowledge fetched for the given sample index (k, i). The fetched knowledge is ensembled in an average manner, which can be expressed as:

$$\overline{zr}_i^k = \frac{1}{R}\sum_{s=1}^{R}(zr_i^k)_s. \tag{17}$$

Subsequently, the ensembled knowledge is distributed to client k for performing distillation-based local model optimization weighted by factor β, which is defined as follows:

$$\begin{aligned}&\arg\min_{W^k} J^k(W^k)\\ =&\arg\min_{W^k}\sum_{(X_i^k, y_i^k)\in\mathcal{D}^k}[L_{CE}(\tau(f^k(X_i^k)), y_i^k)\\ &+\beta\cdot KL(\tau(f^k(X_i^k))||\tau(\overline{zr}_i^k))].\end{aligned} \tag{18}$$

D. Formal Description of FedCache

The overview of executing procedure of FedCache is shown in Fig. 4, and the execution processes of FedCache on client k and the server are respectively formulated in Algorithms 1 and 2. From the overall perspective, we allow personalized local models on devices to distill ensembled knowledge on the samples similar to private data with the assistance of the server-side knowledge cache.

Specifically, FedCache consists of the following steps:

- *Hash Encoding and Uploading:* For each sample from a given client, a hash value is encoded based on the pre-trained local encoder according to (14), (Algorithm 1, line 3). This hash value is uploaded to the server along with the corresponding label and sample index (Algorithm 1, line 4). As the encoder is a deep pre-trained neural network with a large number of superimposed non-linear mapping and the dimensionality of the output code is much smaller than that of data, sharing hash values with the server is privacy-preserving.
- *Knowledge Cache Initialization:* The server accepts the uploaded information from clients (Algorithm 2, line 3) and establishes relations between sample indexes in the knowledge cache according to (7), (8), (9), (10), and (11), such that each sample can be indexed to R-related samples (Algorithm 2, line 4–6).

Algorithm 1: FedCache on Client k.

1 //Initialization process
2 **foreach** $(X_i^k, y_i^k) \in \mathcal{D}^k$ **do**
3 $h_i^k \leftarrow f^h(X_i^k)$
4 Upload h_i^k with index (k, i) and label y_i^k to the server
5 **end**
6 //Training process
7 **repeat**
8 **foreach** $(X_i^k, y_i^k) \in \mathcal{D}^k$ **do**
9 $z_i^k \leftarrow f^k(X_i^k)$
10 Upload z_i^k with index (k, i) to the server
11 Download averaged ensemble knowledge $\overline{zr}_i^k$ from the server
12 $W^k \leftarrow W^k - lr \cdot \nabla_{W^k} J^k(W^k)$
13 ▷ Optimize Eq. (18)
14 **end**
15 **until** *Training stop*;

Algorithm 2: FedCache on the Server.

1 //Initialization process
2 **repeat**
3 Receive h_i^k with index (k, i) and label y_i^k from client k
4 Update LI, IK, IH according to Eq. (7, 8, 9)
5 **until** *Receive all indexes* (k, i) *from* K *clients*;
6 Build relations via HNSW [43] according to Eq. (10) and Eq. (11)
7 //Training process
8 **repeat**
9 **foreach** (k, i) **do**
10 Receive (k, i) and z_i^k from client k
11 Fetch R related knowledge from the knowledge cache according to Eq. (16) and Eq. (12)
12 Obtain ensembled fetched knowledge $\overline{zr}_i^k$ according to Eq. (17)
13 Send $\overline{zr}_i^k$ to client k
14 Update knowledge cache according to Eq. (13)
15 **end**
16 **until** *Training stop*;

- *Knowledge Extraction and Uploading:* Clients extract logits (Algorithm 1, line 9) and upload logits with corresponding sample indexes to the server (Algorithm 1, line 10). This step is an alternative to the parameters/features uploading step of PIA and SLIA-FE. As the size of the logits and sample indexes are several orders of magnitude smaller than that of the model parameters or features, the communication burden can be significantly reduced.
- *Knowledge Fetching:* The server accepts the sample indexes uploaded by the clients (Algorithm 2, line 10), and fetches R-nearest matching knowledge from the knowledge cache based on pre-established sample index relations (Algorithm 2, line 11). This step enables on-device models to obtain sample-level granularity of knowledge without being limited by the number of classes.
- *Knowledge Ensemble and Distributing:* The fetched knowledge is ensembled on the server according to (17) (Algorithm 2, line 12), and is subsequently distributed to corresponding clients (Algorithm 2, line 13). This step is also communication-efficient since only logits are transferred between the server and clients.
- *Knowledge Update:* The stored knowledge in the knowledge cache is updated based on the newly-uploaded knowledge (in Algorithm 2, line 10) according to (13). (Algorithm 2, line 14)
- *Knowledge Acceptance and Distillation:* The clients receive the ensembled knowledge distributed from the server (Algorithm 1, line 11) and optimize client-side local models according to (18) (Algorithm 1, lines 12–13). This step can be performed asynchronously on each client without waiting for other clients to finish their previous steps.

V. Experiments

A. Experimental Setup

1) Datasets and Preprocessing: We conduct experiments on four common datasets, MNIST [45], FashionMNIST [44], CIFAR-10 [46] and CINIC-10 [47]. Following [22], we adopt the data partitioning scheme in FedML [48], which uses a hyperparameter α ($\alpha > 0$) to control the degree of local data distribution differentiation among devices. As α decreases, the data distributions among devices show greater degrees of heterogeneity. To evaluate FedCache on personalized data, we use the same data partitioning strategy for both the complete training and testing datasets, ensuring that the label distributions of training and testing local data are consistent on each device. In all of our main experiments, we partition each dataset into 300 non-independent identically distributed copies for training and testing on $K = 300$ different clients, and the hyper-parameter α is set to 1.0. Each client runs locally for one epoch before model aggregation or feature/knowledge transfer.

2) Benchmarks and Criteria: To fully demonstrate the effectiveness of FedCache, we compare it with the state-of-art PFL methods with various architectures, including FMTL [13] and pFedMe [12] based on PIA, FedDKC [24] and FedICT [22] based on SLIA-FE, and FD [23] based on CLIA. Among all the architectures, SLIA-PD is discarded because of its impractical reliance on public datasets. The precision of benchmark algorithms is measured by Maximum Average User model Accuracy [13] (MAUA). Moreover, we denote the communication overhead to reach a given average UA acc as $acc@$, measuring system communication efficiency with different $acc@$ according to the actual system performance, as shown in Table III. We also calculate the speed-up ratio of each method by comparing the ratio of communication overhead between the one with the highest communication overhead of all benchmark algorithms and this method under the same experimental settings. In addition, our

TABLE III
acc@ USED IN DIFFERENT EXPERIMENTS TO MEASURE SYSTEM COMMUNICATION OVERHEAD

Model Setting	Dataset			
	MNIST	Fashion MNIST	CIFAR-10	CINIC-10
Model Homo.	87@	77@	43@	40@
Model Hetero.	83@	77@	41@	41@

TABLE IV
MAIN CONFIGURATIONS OF FOUR ADOPTED MODELS

Model	Notation	Feat. Shape	Params
ResNet-small	A_1^C	$H \times W \times 16$	76.2K
ResNet-medium	A_2^C		171.2K
ResNet-large	A_3^C		266.1K
ResNet-server	A^S		588.2K

The height and width of the input images are noted as *H* and *W*, respectively.

MAUA results are obtained in a reasonable training time, when the algorithm reaches convergence or the total communication overhead reaches the given limitation, such as 55 G and 19 G for CIFAR-10 and CINIC-10 datasets, respectively.

3) Models: For the deep pre-trained encoder, we adopt MobileNetV3 [49] pre-trained on ImageNet [50], with the last fully connected layer removed. In addition, we consider 4 different model architectures, where $\{A_1^C, A_2^C, A_3^C\}$ are for clients, and A^S is for the server, and the main configurations of four adopted models are shown in Table IV. It is worth noting that the model on the server does not contain the foremost Conv+Batch+ReLU layers to fit the training requirements of [24], [25]. Moreover, both client-side model homogeneity and heterogeneity are considered in our experiments. Specifically, for the experiments with homogeneous models, we compare FedCache with all aforementioned benchmark algorithms, and all clients adopt the model architecture A_3^C. For the experiments with heterogeneous models, FedCache only compares with the benchmarks that support model heterogeneity among clients, including FedDKC, FedICT and FD, and clients with residuals of index mod 3 of 0, 1 and 2 are assigned with model architectures A_1^C, A_2^C and A_3^C respectively.

4) Hyper-Parameter Settings: We adopt stochastic gradient descent with a learning rate $lr = 0.01$ and a batch size of 8 in all the experiments. In addition, the hyper-parameters of benchmark algorithms are set as follows:

- For pFedMe, we set $\eta = 0.005$, $\lambda = 15$ and $\beta = 1$ according to [12].
- For MTFL, we adopt the FedAvg optimization strategy [13], with other hyper-parameters following the default setting in [51].
- For FedDKC, we adopt KKR as the knowledge refinement strategy, with $\beta = 1.5$ and $T = 0.12$ according to [24].
- For FedICT, we adopt the similarity-based LKA strategy, with $\beta = \lambda = \mu = 1.5$ and $T = 3.0$ according to [22].
- For FD, no individualized hyper-parameters are required [23].

Finally, for our proposed FedCache, we set $\beta = 1.5$ and $R = 16$. The impact of hyper-parameters on system performance will be investigated in the ablation study.

B. Results

1) Performance on Homogeneous Models: Table V displays MAUA and communication overhead of FedCache compared to all considered benchmarks on different datasets, and the MAUA performance per unit of communication overhead is shown in Fig. 5. As can be seen from Table V, FedCache achieves 77.71%, 44.42%, and 40.45% MAUA on FashionMNIST, CIFAR-10, and CINIC-10 datasets, respectively, which are is comparable to the considered benchmark algorithms. Meanwhile, according to the criteria described in V-A2, the total communication overhead of FedCache over the three datasets mentioned above are all less than 0.20 G, and the speed-up ratios of FedCache are all over $\times 78$, which enable the communication efficiency to be much higher than existing methods with previous architectures. This is because FedCache adopts a lightweight communication protocol with only logits and hash values being transferred, and does not transmit model parameters as well as features with relatively large sizes. Moreover, the efficient communication of FedCache can be further verified in Fig. 5, where our method exhibits a much steeper convergence curve than FedDKC, FedICT, pFedMe, and MTFL. We can also observe in Fig. 5 that compared to communication-efficient FD, FedCache achieves significantly higher MAUA, achieving satisfactory system performance while maintaining communication efficiency orders of magnitude higher than other benchmark algorithms. The reason is that FedCache is an SLIA architecture rather than CLIA, where enriched knowledge can be utilized to obtain significantly more information for on-device model constructive optimization, thus possessing performance superiority.

2) Performance on Heterogeneous Models: Table VI shows the comparison of FedCache with benchmark algorithms that support model heterogeneity on clients. Likewise, we can conclude that FedCache achieves comparable MAUA to considered benchmarks, but with extremely high communication efficiency due to the elimination of feature transfer compared with FedDKC and FedICT. Similar to section V-B1, FedCache obtains a convergence curve in Fig. 6 that is capped above FD, which indicates that FedCache gains better system performance than FD. This further confirms the superiority of our proposed FedCache architecture over heterogeneous models.

VI. ABLATION STUDY

A. Ablation Settings

In this section, we conduct the ablation study to investigate the impact of three factors on the performance of FedCache: the degree of data heterogeneity, the proportion of local samples, and the number of related samples. All ablation experiments are evaluated on the FashionMNIST dataset, with the same settings adopted for experiments with homogeneous models in section

TABLE V
MAUA (%), COMMUNICATION OVERHEAD AND COMMUNICATION EFFICIENCY SPEED-UP RATIO ON HOMOGENEOUS ON-DEVICE MODELS

Dataset	Method	Model		Metric		
		Client	Server	MAUA (%)	Comm. (G)	Speed-up Ratio
MNIST	pFedMe	A_3^C	A_3^C	94.89	13.25	×1.0
	MTFL			95.59	7.77	×1.7
	FedDKC	A_3^C	A^S	89.62	9.13	×1.5
	FedICT			84.62	-	-
	FD	A_3^C	-	84.19	-	-
	FedCache			87.77	**0.99**	**×13.4**
FashionMNIST	pFedMe	A_3^C	A_3^C	81.57	20.71	×1.0
	MTFL			83.92	12.33	×1.7
	FedDKC	A_3^C	A^S	78.24	8.43	×2.5
	FedICT			76.90	13.34	×1.6
	FD	A_3^C	-	76.32	-	-
	FedCache			77.71	**0.08**	**×258.9**
CIFAR-10	pFedMe	A_3^C	A_3^C	37.49	-	-
	MTFL			43.43	52.99	×1.0
	FedDKC	A_3^C	A^S	45.87	11.46	×4.6
	FedICT			43.61	10.69	×5.0
	FD	A_3^C	-	42.77	-	-
	FedCache			44.42	**0.19**	**×278.9**
CINIC-10	pFedMe	A_3^C	A_3^C	31.65	-	-
	MTFL			34.09	-	-
	FedDKC	A_3^C	A^S	43.95	4.12	×1.3
	FedICT			42.79	5.50	×1.0
	FD	A_3^C	-	39.36	-	-
	FedCache			40.45	**0.07**	**×78.6**

Some methods are unable to calculate the communication overhead with corresponding speed-up ratios as they cannot achieve the MAUA in TABLE 3 under given experimental settings, and their corresponding items are denoted by -. The same as below.

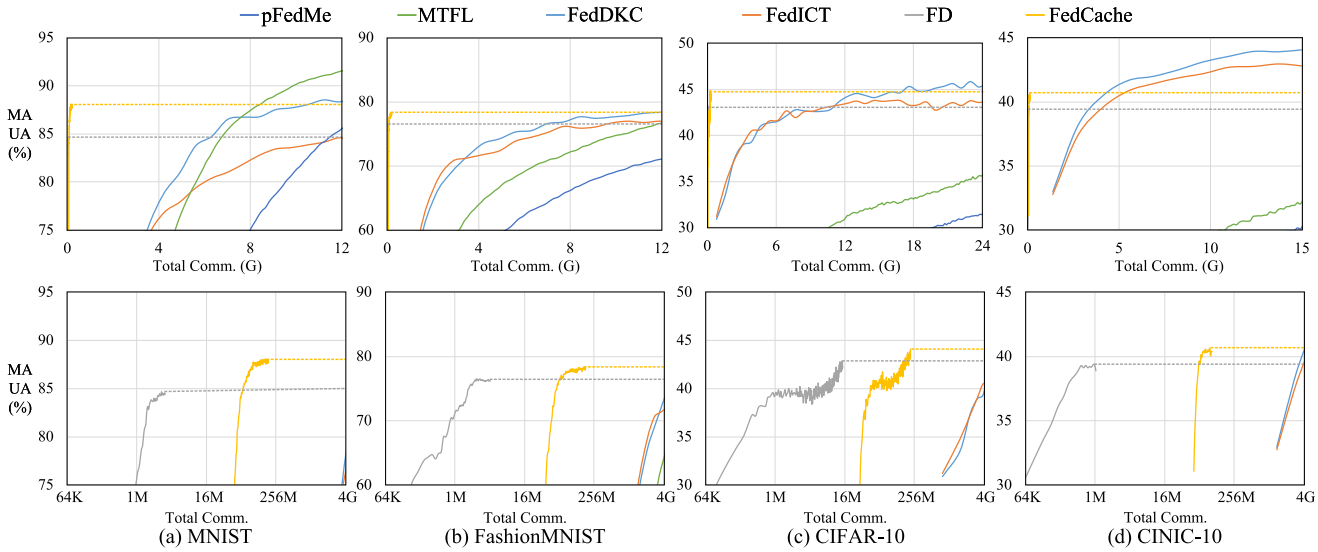

Fig. 5. MAUA (%) per unit of communication overhead in experiments with homogeneous models. Dashed lines indicate the extension of algorithms beyond convergence to the maximum MAUA over communication overheads. The same as below.

TABLE VI
MAUA (%), COMMUNICATION OVERHEAD AND COMMUNICATION EFFICIENCY SPEED-UP RATIO ON HETEROGENEOUS ON-DEVICE MODELS

Dataset	Method	Model		Metric		
		Client	Server	MAUA (%)	Comm. (G)	Speed-up Ratio
MNIST	FedDKC	A_1^C, A_2^C, A_3^C	A^S	85.38	10.53	×1.0
	FedICT			80.53	-	-
	FD		-	79.90	-	-
	FedCache			83.94	**0.10**	**×105.3**
FashionMNIST	FedDKC	A_1^C, A_2^C, A_3^C	A^S	77.96	12.64	×1.0
	FedICT			76.11	-	-
	FD		-	75.57	-	-
	FedCache			77.26	**0.08**	**×158.0**
CIFAR-10	FedDKC	A_1^C, A_2^C, A_3^C	A^S	44.53	4.58	×1.2
	FedICT			43.96	5.35	×1.0
	FD		-	40.40	-	-
	FedCache			41.59	**0.05**	**×107.0**
CINIC-10	FedDKC	A_1^C, A_2^C, A_3^C	A^S	44.80	4.12	×1.3
	FedICT			43.40	5.50	×1.0
	FD		-	40.76	-	-
	FedCache			41.71	**0.07**	**×78.6**

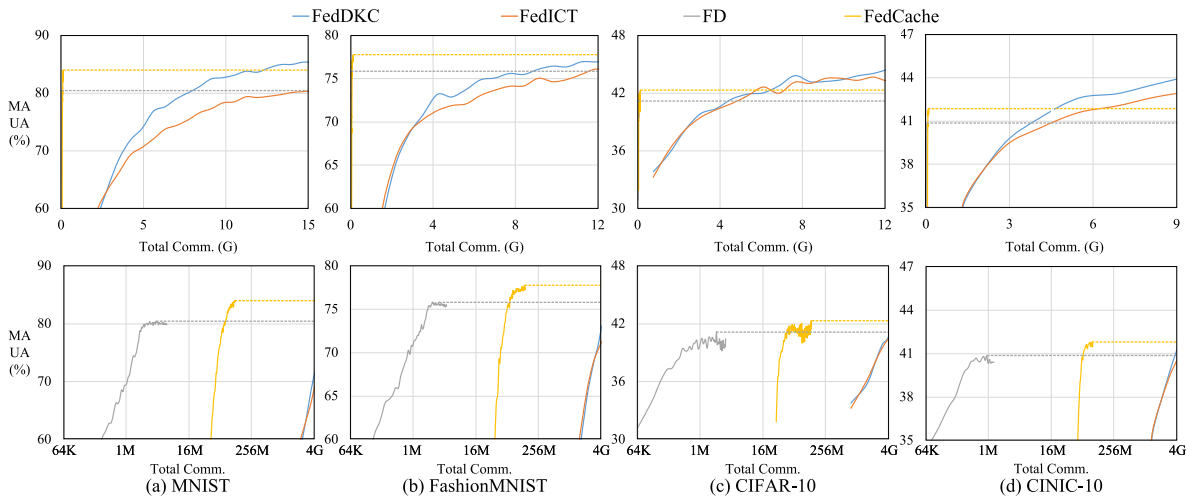

Fig. 6. MAUA (%) per unit of communication overhead in experiments with heterogeneous models.

V-B1 by default. The performance of all algorithms is measured by MAUA (%) in the following subsections.

B. Results

1) Impact of Degree of Data Heterogeneity: To explore the effect of data heterogeneity on the performance of FedCache, we set the hyper-parameter α to different values $\alpha \in \{1.0, 3.0, 10.0\}$ to control the degree of data heterogeneity, and compare the performance of FedCache with FD, with the results shown in Table VII and Fig. 7(a). It can be seen that FedCache consistently outperforms FD despite the skewness of local data distributions, reflecting the adaptability of our method to different data environments.

TABLE VII
PERFORMANCE OF FEDCACHE AND FD WITH DIFFERENT α

Method	Degree of Data Heterogeneity		
	α=1.0	α=3.0	α=10.0
FD	76.32	71.92	70.67
FedCache	**77.71**	**72.92**	**71.73**

2) Impact of Local Data Proportion: To investigate the performance of FedCache with different percentages of local data to the overall data, we control the number of different local samples to $\{0.33\%, 1\%, 5\%, 20\%\}$ of the whole dataset, and compare the performance of FedCache with FD, with the results shown in Table VIII and Fig. 7(b). We can observe that the performance

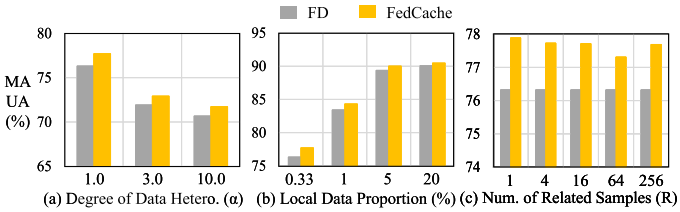

Fig. 7. Impact of the degree of data heterogeneity, local data proportion, and the number of related samples on the performance of FedCache.

TABLE VIII
PERFORMANCE OF FEDCACHE AND FD WITH DIFFERENT AVERAGE LOCAL SAMPLE PROPORTIONS

Method	Average Local Sample Proportion			
	0.33%	1%	5%	20%
FD	76.32	83.41	89.37	90.07
FedCache	**77.71**	**84.32**	**90.03**	**90.47**

TABLE IX
PERFORMANCE OF FEDCACHE AND FD WITH DIFFERENT R

Method	MAUA(%)
FD	76.32
FedCache (R=1)	77.89
FedCache (R=4)	77.73
FedCache (R=16)	77.71
FedCache (R=64)	77.31
FedCache (R=256)	77.69

of both FD and FedCache improves as the local sample share of clients increases. Still, FedCache always outperforms FD, which confirms the superior performance of our FedCache with varying percentages of local data from a single client.

3) Impact of Number of Related Samples: To evaluate the performance of FedCache with different numbers of related samples, we set $R \in \{1, 4, 16, 64, 256\}$ and compare the performance of FedCache with FD with the aforementioned R settings, with the results shown in Table IX and Fig. 7(c). It can be seen that our method consistently outperforms FD in different R settings. This indicates that FedCache is robust to the choice of related samples and can achieve satisfactory performance consistently.

VII. DISCUSSION

A. Analysis on Computation Complexity

We compare the communication complexity of FedCache with other PFL architectures in Table X. On the device side, FedCache has the identical computation complexity as PIA, SLIA-FE and CLIA, since their computation all mainly focuses on on-device models' forward propagation on local data. As there is no need for local training based on public datasets, FedCache has lower computation complexity on the device side compared to SLIA-PD. On the server side, the computation overhead of FedCache mainly consists of establishing relations among samples through R-nearest neighbors retrieval and integrating relevant knowledge of a given sample index in each round. The server-side computation complexity comparison between FedCache and PIA depends on the scale of model parameters and the average number of local samples per client. In our experiments on four datasets with $R = 16$, $r > 100$, hundreds of samples held on a single client and the parameter size $> 50\,K$, the server-side computation complexity of FedCache is much smaller than that of PIA. Since FedCache converges better than PIA in empirical experiments, the superiority of computation overhead of FedCache over PIA will be pronounced in reality. In addition, we claim that the server-side computation complexity of FedCache is much smaller than that of SLIA-FE and SLIA-PD. The reason for the former is that FedCache doesn't require forward propagation on server-side model training for each sample. While the reason for the latter is that the average size of private data per client is much smaller than that of public datasets in practice. Although CLIA achieves relatively-low server-side computation complexity over FedCache, it pays the price of significantly reduced knowledge enrichment, resulting in poor performance confirmed by empirical experiments in section V, so FedCache still possesses an irreplaceable superiority over CLIA in terms of balancing computation overhead and system performance.

B. Limitations

We analyze that the limitations of FedCache are threefold. One limitation of FedCache is that it conducts knowledge distillation on device-side models only based on knowledge associated with local samples, but neglects knowledge learning for global generalization. As a result, it is only suitable for personalization tasks rather than general tasks that require global generalization capabilities. The generalization performance of FedCache can be improved when introducing additional information, such as partial global parameters or global public data. Another limitation is that we apply our method only to conventional image classification problems in our experiments, and additional research on data encoding strategies, hash correlation measures for serialized data and other non-image structured data are also meaningful for FedCache. In addition, FedCache cannot support PFL with dynamism and continuity data, while end devices may continuously generate new data that requires real-time processing and analysis [52]. By considering and addressing the above limitations, FedCache can further enhance its effectiveness in a wider range of applications.

VIII. CONCLUSION

In this paper, we propose FedCache, a novel federated learning architecture tailored for personalized edge intelligence. FedCache designs a knowledge cache on the server for storing newly-extracted knowledge uploaded by clients and fetching correlatively personalized knowledge from samples with similar hashes to the specified private data. On this basis, ensemble distillation is performed on device-side local models for personalized constructive optimization. To our best knowledge,

TABLE X
COMPARISON OF THE COMPUTATION COMPLEXITY OF PFL ARCHITECTURES

	PIA	SLIA-FE	SLIA-PD	CLIA	FedCache
End Devices	$rN^k \cdot \mathrm{O}(W^k)$	$rN^k \cdot \mathrm{O}(W^k)$	$r(N^k + N^P) \cdot \mathrm{O}(W^k)$	$rN^k \cdot \mathrm{O}(W^k)$	$rN^k \cdot \mathrm{O}(W^k)$
Edge Server	$rK \cdot \mathrm{O}(W^S)$	$r \sum_{k=1}^{K} N^k \cdot \mathrm{O}(W^S)$	$rN^P K \cdot \mathrm{O}(C)$	$rK \cdot \mathrm{O}(C^2)$	$R \sum_{k=1}^{K} N^k \log \sum_{k=1}^{K} N^k \cdot \mathrm{O}(F)$ $+rR \sum_{k=1}^{K} N^k \cdot \mathrm{O}(C)$

r represents the total communication rounds. N^P represents the number of samples in the public dataset. F represents the scale of transmitted features.

FedCache is the first architecture for personalized federated learning that enables sample-grained logits interaction without features transmission or public datasets. Empirical experiments show that FedCache achieves comparable accuracy with state-of-the-art personalized federated learning methods with various architectures, meanwhile reducing communication costs by two orders of magnitude.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Anfu Zhou from Beijing University of Posts and Telecommunications, Tian Wen, Quyang Pan, Xujing Li, Chungang Lin from the Institute of Computing Technology, Chinese Academy of Sciences, and Yuhan Tang, Aoxu Zhang from Beijing Jiaotong University for inspiring suggestions.

REFERENCES

[1] D. Xu et al., "Edge intelligence: Empowering intelligence to the edge of network," *Proc. IEEE*, vol. 109, no. 11, pp. 1778–1837, Nov. 2021.
[2] B. Yang, B. Wu, Y. You, C. Guo, L. Qiao, and Z. Lv, "Edge intelligence based digital twins for Internet of Autonomous Unmanned Vehicles," *Softw.: Pract. Experience*, 2022, doi: 10.1002/spe.3080
[3] M. Nasir, K. Muhammad, A. Ullah, J. Ahmad, S. W. Baik, and M. Sajjad, "Enabling automation and edge intelligence over resource constraint iot devices for smart home," *Neurocomputing*, vol. 491, pp. 494–506, 2022.
[4] Y. Gong et al., "Edgerec: Recommender system on edge in mobile taobao," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 2477–2484.
[5] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–23, 2022.
[6] Q. Liu, S. Sun, M. Liu, Y. Wang, and B. Gao, "Online spatio-temporal correlation-based federated learning for traffic flow forecasting," 2023, *arXiv:2302.08658*.
[7] R. Kanagavelu et al., "Federated learning for advanced manufacturing based on industrial IoT data analytics," in *Implementing Industry 4.0: The Model Factory as the Key Enabler for the Future of Manufacturing*, Springer, 2021, pp. 143–176.
[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR 2017, pp. 1273–1282.
[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.
[10] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *Proc. 4th World Conf. Smart Trends Syst., Secur. Sustainability*, 2020, pp. 794–797.
[11] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, Dec. 2023.
[12] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with Moreau envelopes," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 21394–21405, 2020.
[13] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 630–641, Mar. 2022.
[14] H. Jin et al., "Personalized edge intelligence via federated self-knowledge distillation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 2, pp. 567–580, Feb. 2023.
[15] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, "Communication-efficient federated learning via knowledge distillation," *Nature Commun.*, vol. 13, no. 1, pp. 1–8, 2022.
[16] F. Sattler, A. Marban, R. Rischke, and W. Samek, "CFD: Communication-efficient federated distillation via soft-label quantization and delta coding," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2025–2038, Jul./Aug. 2022.
[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[18] Z. Wu et al., "Spirit distillation: A model compression method with multi-domain knowledge transfer," in *Proc. Knowl. Sci., Eng. Manage.: 14th Int. Conf.*, Tokyo, Japan, Springer, 2021, pp. 553–565.
[19] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, 2021.
[20] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu, "Parameterized knowledge transfer for personalized federated learning," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 10092–10104, 2021.
[21] Y. J. Cho, J. Wang, T. Chirvolu, and G. Joshi, "Communication-efficient and model-heterogeneous personalized federated learning via clustered knowledge transfer," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 234–247, Jan. 2023.
[22] Z. Wu et al., "FediCT: Federated multi-task distillation for multi-access edge computing," *IEEE Trans. Parallel Distrib. Syst.*, early access, Jun. 26, 2023, doi: 10.1109/TPDS.2023.3289444.
[23] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data," 2018, *arXiv: 1811.11479*.
[24] Z. Wu et al., "Exploring the distributed knowledge congruence in proxy-data-free federated distillation," *ACM Trans. Intell. Syst. Technol.*, Dec. 2023. [Online]. Available: https://doi.org/10.1145/3639369
[25] Z. Wu, S. Sun, Y. Wang, M. Liu, X. Jiang, and R. Li, "Survey of knowledge distillation in federated edge learning," 2023, *arXiv:2301.05849*.
[26] B. Tan, B. Liu, V. Zheng, and Q. Yang, "A federated recommender system for online services," in *Proc. 14th ACM Conf. Recommender Syst.*, 2020, pp. 579–581.
[27] X. Zhou, Y. Tian, and X. Wang, "Source-target unified knowledge distillation for memory-efficient federated domain adaptation on edge devices," 2022. [Online]. Available: https://openreview.net/forum?id=8rCMq0yJMG
[28] H. Jiang, M. Liu, B. Yang, Q. Liu, J. Li, and X. Guo, "Customized federated learning for accelerated edge computing with heterogeneous task targets," *Comput. Netw.*, vol. 183, 2020, Art. no. 107569.
[29] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-IID private data," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 191–205, Jan. 2023.
[30] L. Zhang, D. Wu, and X. Yuan, "FedZKT: Zero-shot knowledge transfer towards resource-constrained federated learning with heterogeneous on-device models," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst.*, 2022, pp. 928–938.
[31] S. Yu, W. Qian, and A. Jannesari, "Resource-aware federated learning using knowledge extraction and multi-model fusion," 2022, *arXiv:2208.07978*.

[32] A. Tak and S. Cherkaoui, "Federated edge learning: Design issues and challenges," *IEEE Netw.*, vol. 35, no. 2, pp. 252–258, Mar./Apr. 2021.
[33] R. Yu and P. Li, "Toward resource-efficient federated learning in mobile edge computing," *IEEE Netw.*, vol. 35, no. 1, pp. 148–155, Jan./Feb. 2021.
[34] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.
[35] C. J. Hoofnagle, B. Van DerSloot, and F. Z. Borgesius, "The European Union general data protection regulation: What it is and what it means," *Inf. Commun. Technol. Law*, vol. 28, no. 1, pp. 65–98, 2019.
[36] J. Nguyen et al., "Federated learning with buffered asynchronous aggregation," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2022, pp. 3581–3607.
[37] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv: 1903.03934*.
[38] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *Proc. Int. Conf. Learn. Representations*, 2021.
[39] 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.
[40] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large CNNs at the edge," in *Proc. Adv. Neural Inf. Process. Syst.* 80, 2020, pp. 14068–140.
[41] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," 2019, *arXiv: 1910.03581*.
[42] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Communication-efficient federated distillation with active data sampling," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 201–206.
[43] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, Apr. 2020.
[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv: 1708.07747*.
[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[46] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," 2009.
[47] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "CINIC-10 is not imagenet or CIFAR-10," 2018, *arXiv: 1810.03505*.
[48] C. He et al., "FedML: A research library and benchmark for federated machine learning," 2020, *arXiv: 2007.13518*.
[49] A. Howard et al., "Searching for mobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.
[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[51] J. Mills, J. Hu, and G. Min, 2022. [Online]. Available: https://github.com/JedMills/MTFL-For-Personalised-DNNs
[52] C. Savaglio and G. Fortino, "A simulation-driven methodology for IoT data mining based on edge computing," *ACM Trans. Internet Technol.*, vol. 21, no. 2, pp. 1–22, 2021.

Zhiyuan Wu (Member, IEEE) is currently a research assistant with the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). He has contributed several technical papers to top-tier conferences and journals as the first author in the fields of computer architecture, computer networks, and intelligent systems, including *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, *IEEE Transactions on Mobile Computing (TMC)*, *IEEE International Conference on Computer Communications (INFOCOM)*, and *ACM Transactions on Intelligent Systems and Technology (TIST)*. He has served as a technical program committee member or a reviewer for more than 10 conferences and journals, and was invited to serve as a session chair for the International Conference on Computer Technology and Information Science (CTIS). He is a member of ACM, the China Computer Federation (CCF), and is granted the President Special Prize of ICT, CAS. His research interests include federated learning, mobile edge computing, and distributed systems.

Sheng Sun received the bachelor's degree from Beihang University, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences. She is currently an associate professor with the Institute of Computing Technology, Chinese Academy of Sciences. She has led or executed 5 major funded research projects and published more than 20 technical papers in journals and conferences related to computer network and distributed systems, including *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, *IEEE Transactions on Mobile Computing (TMC)*, and *IEEE International Conference on Computer Communications (INFOCOM)*. Her research interests include federated learning, edge intelligence, and privacy computing.

Yuwei Wang (Member, IEEE) received the PhD degree in computer science from the University of Chinese Academy of Sciences, Beijing, China. He is currently an associate professor with the Institute of Computing Technology, Chinese Academy of Sciences. He has been responsible for setting more than 30 international and national standards, and also holds various positions in both international and national industrial standards development organizations (SDOs) as well as local research institutions, including the associate rapporteur with the ITU-T SG16 Q5, and the deputy director of China Communications Standards Association (CCSA) TC1 WG1. His current research interests include federated learning, mobile edge computing, and next-generation network architecture.

Min Liu (Senior Member, IEEE) received the BS and MS degrees in computer science from Xi'an Jiaotong University, China, and the PhD degree in computer science from the Graduate University of the Chinese Academy of Sciences, China. She is currently a professor with the Institute of Computing Technology, Chinese Academy of Sciences, and also holds a position with Zhongguancun Laboratory. Her current research interests include mobile computing and edge intelligence.

Ke Xu (Fellow, IEEE) received the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He serves as a full professor with Tsinghua University. He has published more than 200 technical papers and holds 11 U.S. patents in the research areas of next-generation internet, blockchain systems, the Internet of Things, and network security. He is a member of ACM. He served as the Steering Committee Chair for IEEE/ACM IWQoS. He has guest-edited several special issues in IEEE and Springer journals. He is the editor of *IEEE Internet of Things Journal*.

Wen Wang received the bachelor's degree with honor from Xidian University. He is currently working toward the master's degree with the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include network security, in-network computing and information retrieval.

Xuefeng Jiang received the bachelor's degree with honor from the Beijing University of Posts and Telecommunications. He is currently working toward the PhD degree with the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include distributed optimization and machine learning.

Bo Gao (Member, IEEE) received the MS degree in electrical engineering from the School of Electronic Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai, China, in 2009, and the PhD degree in computer engineering from the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, USA, in 2014. He was an assistant professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China from 2014 to 2017. He was a visiting researcher with the School of Computing and Communications, Lancaster University, Lancaster, U.K. from 2018 to 2019. He is currently an associate professor with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing. He has directed a number of research projects sponsored by the National Natural Science Foundation of China (NSFC) or other funding agencies. He is a member of ACM and China Computer Federation (CCF). His research interests include wireless networking, mobile/edge computing, multiagent systems, and machine learning.

Jinda Lu received the bachelor's degree with honor from Jilin University. He is currently working toward the master's degree with the School of Information Science and Technology, University of Science and Technology of China. His research interests include artificial intelligence, pattern recognition, and knowledge distillation.