Handling Data Heterogeneity for IoT Devices in Federated Learning: A Knowledge Fusion Approach

Xu Zhou, Xinyu Lei, *Member, IEEE*, Cong Yang, Yichun Shi, Xiao Zhang, and Jingwen Shi

***Abstract*— Federated learning (FL) supports distributed training of a global machine learning model across multiple Internet of Things (IoT) devices with the help of a central server. However, data heterogeneity across different IoT devices leads to the client model drift issue and results in model performance degradation and poor model fairness. To address the issue, we design Federated learning with global-local Knowledge Fusion (FedKF) scheme in this paper. The key idea in FedKF is to let the server return the global knowledge to be fused with the local knowledge in each training round so that the local model can be regularized towards the global optima. Therefore, the client model drift issue can be mitigated. In FedKF, we first propose the active-inactive model aggregation technique that supports a precise global knowledge representation. Then, we propose a data-free knowledge distillation (KD) approach to enable each client model to learn the global knowledge (embedded in the global model) while each client model can still learn the local knowledge (embedded in the local dataset) simultaneously, thereby realizing the global-local knowledge fusion process. The theoretical analysis and intensive experiments demonstrate the superiority of FedKF over previous solutions.**

***Index Terms*—Federated learning, IoT, data heterogeneity, knowledge distillation.**

I. Introduction

FEDERATED learning (FL) supports distributed training of a global machine learning model across multiple Internet of Things (IoT) devices with the help of a central server. The local dataset held by each IoT device is never exchanged in FL, so the local dataset privacy is protected. The most famous FL algorithm is FedAvg [1]. In one round of FedAvg training, a central server sends the global model weight to a portion of distributed IoT devices (i.e., active clients). Then, each IoT device trains the model using the local data. Next, each device sends the new model weight to the central server, which is responsible for computing the new aggregated averaged model. Afterward, the server sends the new global model to some re-selected active clients to start the next round of FedAvg training. After numerous rounds of training, the ML model can be well trained. Each device can exploit the well-trained ML model for different applications. In this paper, we refer to IoT devices as clients in FL.

In recent years, FL has been intensively studied in both academia and industry [2]–[6]. One problem rooted in FL is that the heterogeneity of data across different clients can lead to significant model performance (i.e., test accuracy) degradation [7]–[9]. For example, multiple hospitals would like to collaboratively train a disease diagnosis ML model via FL. Each hospital collects patient data independently, so their datasets are unbalanced and non-IID (i.e., heterogeneous). One hospital might have few/no data samples belonging to a certain disease class. When each hospital trains the model locally, its local objective may be far from the global objective. Thus, the averaged global model can be away from the global optima. This phenomenon is called client model drift in some literatures [10]–[13]. The client model drift may lead to poor global model performance.

Except for poor model performance, data heterogeneity may also lead to poor model fairness. Consider a disease diagnosis ML model that is trained by multiple different hospitals via FL. These hospitals use different Internet of Medical Things (IoMT) devices to collect patient data in different geographical areas with different race populations. The FL-trained ML model can achieve high averaged model performance, but it may have large model performance variance across different hospitals (when testing on their local patient datasets). If so, the ML model has biases against certain geographical areas (i.e., geographical discrimination) and race populations (i.e., race discrimination). To mitigate such a model bias issue, the FL-trained ML model should also achieve high fairness, which can be measured by the degree of uniformity in model performance across different clients. The considered concept of fairness is also named accuracy parity [14].

To handle data heterogeneity in FL, several previous solutions have been developed. These solutions can be roughly divided into three categories: 1) model performance-based solutions [10]–[13], [15], [16], 2) multiple-objective optimization-based solutions [14], [17], and 3) personalized FL solutions [18]–[20]. For the model performance-based solutions, they merely consider improving the model accuracy in heterogeneous FL, so the fairness issue is not well addressed. As to the multiple-objective optimization-based solutions, they aim to optimize multiple objectives such as model performance, fairness, robustness, etc. However, they usually trade off model performance for higher fairness, so the model performance in these solutions is worse than FedAvg. Personalized FL solutions work on the post-model-training phase (i.e., model

This work was supported in part by the Collaborative Innovation Major Project of Zhengzhou under grant 20XTZX06013 and in part by the National Science Foundation under Grant CNS-2153393 (Corresponding author: Cong Yang).

Xu Zhou and Cong Yang are with the School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou, Henan 450002, China (e-mail: 202022332015352@gs.zzu.edu.cn; wangyuanyc@zzu.edu.cn).

Xinyu Lei is with the Department of Computer Science, Michigan Technological University, Houghton, MI 49931, USA (e-mail: xinyulei@mtu.edu)

Yichun Shi and Jingwen Shi are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA (e-mail: shiyichu@msu.edu; shijingw@msu.edu)

Xiao Zhang is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, 27708, USA (email: xiao.zhang@duke.edu)

TABLE I
COMPARISON AMONG DIFFERENT PREVIOUS SOLUTIONS AND (●: "YES", ○: "NO").

Solutions		Over FedAvg Model Performance	Fairness-Aware	No Proxy Data Required	No Additional Info. Leakage
Model performance-based	FedAvg [1]	○	○	●	●
	FedGen [12]	●	○	●	○
	CCVR [15]	●	○	●	○
	FedDF [11]	●	○	○	●
	FedProx [10]	●	○	●	●
	MOON [16]	●	○	●	●
	FedGKD [13]	●	○	●	●
Multiple-objective optimization-based	q-FFL [14]	○	●	●	●
	FedMGDA+ [17]	○	●	●	●
FedKF (ours)		●	●	●	●

adaption phase); thus, they are a complementary approach to the solution proposed in this paper. Consequently, we aim to design a privacy-preserving FL scheme (that works in the global model training phase) that achieves both high (i.e., over FedAvg) model performance and high fairness for IoT devices in heterogeneous FL.

Observing the limitations of the previous solutions, a question naturally arises: in heterogeneous FL, is it possible to design a privacy-preserving FL scheme (works in the global model training phase) that achieves a better model performance than FedAvg while still keeping fairness as high as possible? To answer the question, we design **Fed**erated learning with global-local **K**nowledge **F**usion (FedKF) scheme. FedKF does not need to trade off model performance (to below FedAvg) for fairness while still achieving high model fairness and a better model performance than FedAvg. Therefore, FedKF yields a positive answer "Yes" to the above question. The key idea in FedKF is to let the server return global knowledge to shepherd the local training so that all local models will be regularized towards the global optima, thereby reducing the client model drift issue in each training round. The global knowledge shepherded local training process is also named the global-local knowledge fusion process in this paper. In FedKF, two major technical challenges should be well addressed.

The first technical challenge is how to represent global knowledge in each training round. To address this issue, we design T1 (active-inactive model aggregation technique) to generate a model representing global knowledge. In each training round, T1 aggregates not only the active clients' model weights but also the inactive clients' cached model weights obtained in the previous training rounds. Thus, T1 supports a more precise global knowledge representation.

The second technical challenge is how to use global knowledge to shepherd the local training in a privacy-preserving manner. To tackle this challenge, we develop T2 (global-local knowledge fusion technique) to enable each local model to learn both global and local knowledge (embedded in each local dataset). Specifically, we use knowledge distillation to transfer the global knowledge from the T1-aggregated model (teacher model) to each local model (student model) during the local training process. However, knowledge distillation on a local dataset is hard to accurately transfer global knowledge due to the inconsistency in data distribution between the local and global datasets. To address this issue, in T2, FedKF lets each client train a local generator to generate imitated samples that follow the distribution of the global dataset. Thus, FedKF can facilitate knowledge distillation using the generated samples to transfer the global knowledge from the T1-aggregated model to the local model when the local model learns the local knowledge, realizing the global-local knowledge fusion process.

The main contributions of this paper are summarized as follows.

- We make the first step forward to design a privacy-preserving FL scheme that achieves both high (i.e., over FedAvg) model performance and high fairness for IoT devices in heterogeneous FL.
- We propose two techniques, T1 (active-inactive model aggregation technique) and T2 (global-local knowledge fusion technique), used in FedKF. Both T1 and T2 can help to improve model performance and model fairness in heterogeneous FL.
- We theoretically prove that FedKF can directly turn out to be a good solution to achieve high model performance and high fairness in heterogeneous agnostic FL. Thus, FedKF has much broader impacts in reality.

The remainder of the paper is organized as follows. Section II reviews the related work, In Section III, we introduce the background knowledge and formal problem statement. In Section IV, the detailed FedKF design is presented. Analysis of FedKF is performed in Section V. In Section VI, we evaluate FedKF and report the experimental results. In Section VII, some conclusions are drawn.

II. RELATED WORK

To handle data heterogeneity in FL, we can develop solutions at two phases of FL: global model training phase and local model adaption phase. In the global model training phase, several solutions have been proposed. These previous solutions can be further classified into two categories: model performance-based solutions and multiple-objective optimization-based solutions. In the local model adaption

phase, personalized FL solutions were developed. These solutions are analyzed as follows.

Global Model Training Phase: Model Performance-based Solutions. These solutions merely consider improving the model accuracy in heterogeneous FL. These solutions include FedGen [12], CCVR [15], FedDF [11], MOON [16], FedProx [10], and FedGKD [13]. In FedGen, the server learns a lightweight generator to ensemble knowledge of local models in a data-free manner, then broadcasts it to clients to regularize the local training. CCVR adjusts the global model using virtual representations sampled from an approximated Gaussian mixture model. FedDF proposes a model fusion method using ensemble distillation where the server fuses the knowledge from local models and transfers it to the global model by using knowledge distillation on unlabeled proxy data. FedGen and CCVR require each client to share additional local dataset information with the server. More specifically, FedGen requires local label count information leaked to the server, while CCVR requires mean and covariance of local features for each class leaked to the server. Compared with FedAvg, these two solutions suffer from additional information leakage, resulting in a security level downgrade. Moreover, compared with FedKF, FedGen needs extra communication overhead. This is because FedGen needs to train a generator on the server with the label count and broadcast the generator to clients in each training round. Compared with FedGen, FedKF employs a more accurate knowledge distillation approach. For knowledge distillation, FedGen uses pseudo features with labels generated by the generator directly to train the predictor of the local model. In FedKF, global knowledge flows from the teacher model to the local model by knowledge distillation with pseudo samples generated by the local generator. As to FedDF, it assumes there are additional proxy data available on the central server for ensemble distillation. Such a strong assumption makes them impractical since the proxy data is unavailable in most cases. FedProx can be viewed as a generalization and re-parametrization of FedAvg. FedProx adds a proximal term to the local subproblem to restrict the local update closer to the initial (global) model. MOON utilizes the similarity between model representations to correct the local training, i.e., conducting contrastive learning at the model level. FedGKD fuses the knowledge from historical global models to guide the local model training where each client learns the global knowledge from past global models via adaptive knowledge distillation techniques. For FedProx, MOON, and FedGKD, their performances are good on less heterogeneous data, but their performances decrease dramatically as data heterogeneity increases. Note that none of the solutions in this category are fairness-aware in their initial design.

Global Model Training Phase: Multiple-objective Optimization-based Solutions. These solutions aim to optimize multiple objectives such as model performance, fairness, robustness, etc. These solutions include q-FFL [14] and FedMGDA+ [17]. Q-FFL reweights the objective—assigning higher weights to clients with poor performance to encourage a more uniform accuracy distribution across clients in FL. FedMGDA+ uses multi-objective optimization to obtain a fairer global model and guarantee that the global model converges to Pareto stationary solutions, refraining from sacrificing the performance of any client. Nevertheless, the fairness gain of q-FFL and FedMGDA+ is obtained by sacrificing model performance, so they cannot achieve a better model performance than FedAvg in theory. For example, when setting the reweighting parameter $q = 0$ in q-FFL, it reduces to FedAvg. If $q > 0$, the model performance of q-FFL is worse than FedAvg.

Local Model Adaption Phase: Personalized FL Solutions. These solutions leverage various strategies to adapt the global model to each local dataset. Some recent solutions include [18]–[20]. A good survey can be found in [21]. Note that personalized FL solutions can be used together with the solutions in the global model training phase to further enhance the performance.

III. Preliminaries and Problem Statement

In this section, we introduce the background knowledge and formally describe the problem statement.

A. Preliminaries

Federated Learning. FL is a distributed machine learning setting where a group of clients jointly train a high-quality centralized model without requiring clients to share their local private data [22]. Suppose that there are K clients jointly to train an ML model in FL. For the k-th client, it stores a local dataset $\mathcal{D}_k$. FL aims to learning a global model weight w over the global dataset $\mathcal{D} = \cup\{\mathcal{D}_k\}_{k=1}^{K}$. Accordingly, the objective of FL is to solve the following optimization problem [1]:

$$\min_{w} f(w) = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} F_k(w), \tag{1}$$

where $F_k(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}_k}[\ell(w; x, y)]$ represents the local objective function at the k-th client.

Knowledge Distillation. Knowledge distillation (KD) technique enables a student model to learn from one or multiple teacher models [23], [24]. KD supports the student model compression while enabling the student model to inherit knowledge distilled from teacher(s). The classic KD techniques (e.g., [11], [23]) require a proxy dataset during the distillation. To eliminate the requirement for the proxy dataset, data-free KD is proposed [25]–[27]. A popular solution for data-free KD is to use the idea of generative adversarial networks (GANs) [26]. A generator is trained to produce imitated training data (to replace the original training dataset) used for KD.

B. Problem Statement

In this paper, each IoT device is referred to as a client. We consider the local datasets unbalanced and non-IID (i.e., heterogeneous). Next, we define the following three metrics to facilitate our problem description.

Definition 1: (**Average Model Performance**) The average model performance (AMP) of model w (denoted as AMP_w) is defined as the average test accuracy of model w on the K clients. Let $a_w^k (k = 1, \ldots, K)$ represent the test accuracy of

model w on k-th client's local test dataset. AMP_w can also be computed as $AMP_w = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} a_w^k$.

Definition 2: (**Fairness Metric**) The fairness metric (FM) of model w (denoted as FM_w) is defined as $FM_w = \mathrm{Var}(a_w^1, \ldots, a_w^K)$, where Var denotes the variance. It is given by $\mathrm{Var} = \frac{1}{K}\sum_{k=1}^{K}(a_w^k - \overline{a}_w)^2$ and $\overline{a}_w = \frac{1}{K}\sum_{k=1}^{K} a_w^k$. A smaller FM_w indicates a fairer model w.

Definition 3: (**Worst-case Local Performance**) The worst-case local performance (WLP) of model w (denoted as WLP_w) is defined as $WLP_w = \min\{a_w^1, \ldots, a_w^K\}$.

WLP as Joint Performance Metric. According to Definitions (1)-(3), the WLP metric can be treated as measuring the joint performance of AMP and FM. On the one hand, given the fixed AMP, a larger WLP tends to indicate a more uniform distribution of local performance (i.e., smaller FM) across clients. On the other hand, given the fixed FM, a larger WLP tends to imply a model with a higher AMP.

Definition 4: (**Privacy-Preserving**) We say an FL scheme is privacy-preserving if it follows the same security principle as FedAvg: for each client, only its model weight can be sent to other entities (e.g., server), and no information about local data can be shared directly.

Design Goals. In this paper, we aim to design a privacy-preserving FL scheme FedKF that can increase AMP while reducing FM. Thus, FedKF can achieve high global performance and fairness simultaneously. If both AMP and FM are jointly considered, FedKF should keep WLP as large as possible. To better explain why WLP can be treated as a joint performance metric, a numerical example is provided. Suppose that there are three models w_1, w_2, and w_3 trained over three clients via FL. Their parameter configurations are shown in Table II. For w_1 and w_2, it holds that $AMP_{w_1} = AMP_{w_2}$. Since $WLP_{w_1} < WLP_{w_2}$, we have $FM_{w_1} > FM_{w_2}$. For w_1 and w_3, it holds that $FM_{w_1} = FM_{w_3}$. Since $WLP_{w_1} < WLP_{w_3}$, we have $AMP_{w_1} < AMP_{w_3}$.

TABLE II
AN NUMERICAL EXAMPLE.

	Client 1	Client 2	Client 3	AMP	FM	WLP
w_1	$a_{w_1}^1 = 0.6$	$a_{w_1}^2 = 0.7$	$a_{w_1}^3 = 0.8$	0.7	0.00667	0.6
w_2	$a_{w_2}^1 = 0.65$	$a_{w_2}^2 = 0.65$	$a_{w_2}^3 = 0.8$	0.7	0.005	0.65
w_3	$a_{w_3}^1 = 0.7$	$a_{w_3}^2 = 0.8$	$a_{w_3}^3 = 0.9$	0.8	0.00667	0.7

Notations. To improve the readability of this paper, we summarize some frequently used notations in Table III.

TABLE III
NOTATIONS.

Notations	Meanings
K	the number of overall clients
m	the number of active clients
C	selection rate of active clients
T	the number of global rounds
E	the number of local epochs
B	local batch size
α	concentration parameter of the Dirichlet distribution
β	the learning rate for training local generator
η	the learning rate for training local model
λ_1	the coefficient of one-hot loss for training generator
λ_2	the coefficient of activation loss for training generator
γ	the coefficient of KL loss for training local model
$\mathcal{D}_k$	the k-th client's local dataset
$\mathcal{D}$	global dataset
AMP	Average Model Performance
FM	Fairness Metric
WLP	Worst-case Local Performance
ACA model	active clients aggregated model
OCA model	overall clients aggregated model
w^t	the ACA model in the t-th round
$\hat{w}^t$	the OCA model in the t-th round
w_k^t	the k-th client's local model in the t-th round
θ_k	the k-th client's local generator

IV. FEDKF DESIGN

In this section, we first overview FedKF and two developed key techniques. Then, we introduce the T2 (global-local knowledge fusion technique) used in FedKF.

A. FedKF & Key Techniques Overview

FedKF. An overview of FedKF is illustrated in Fig. 1. In FedKF, the server maintains K different cache slots for storing the latest local models. In each training round, only the selected active clients need to upload their local models to the server. Thus, the k-th cache slot stores the local model uploaded from k-th client in the most recent training round when k-th client is selected to be active. Informally, FedKF can be described as follows.

- *Step 0*: In the last step of the training round $t-1$, the server aggregates all active clients' uploaded local models to get the active clients aggregated (ACA) model. Meanwhile, FedKF aggregates both active and inactive clients' cached models in the cache slots to get the overall clients aggregated (OCA) model. In this step, active and inactive refer to the clients' state in round $t-1$.
- *Step 1*: In the training round t, a portion of clients are selected as active clients. Let $\{c_1, \ldots, c_m\}$ denote the IDs of the selected clients. Let C represent the selection rate. It follows that $m = C \cdot K$. Then, the server broadcasts the ACA and OCA models to all active clients.
- *Step 2*: On receipt of the two models from the server, each active client treats the ACA model as the local model w_k and treats the OCA model as the teacher model $\hat{w}$.
- *Step 3*: FedKF employs the data-free KD technique to distill the knowledge of the teacher model to the local model. In data-free KD, a generator is trained to generate pseudo-samples to facilitate knowledge transfer. Meanwhile, the local dataset of each active client is used to train the local model. Therefore, both the global knowledge (embedded in the teacher model) and local knowledge (embedded in the local dataset) are fused and transferred to the local model.
- *Step 4*: After global-local knowledge fusion, all active clients upload their local models to the server. Each client's local model serves as the latest local model.

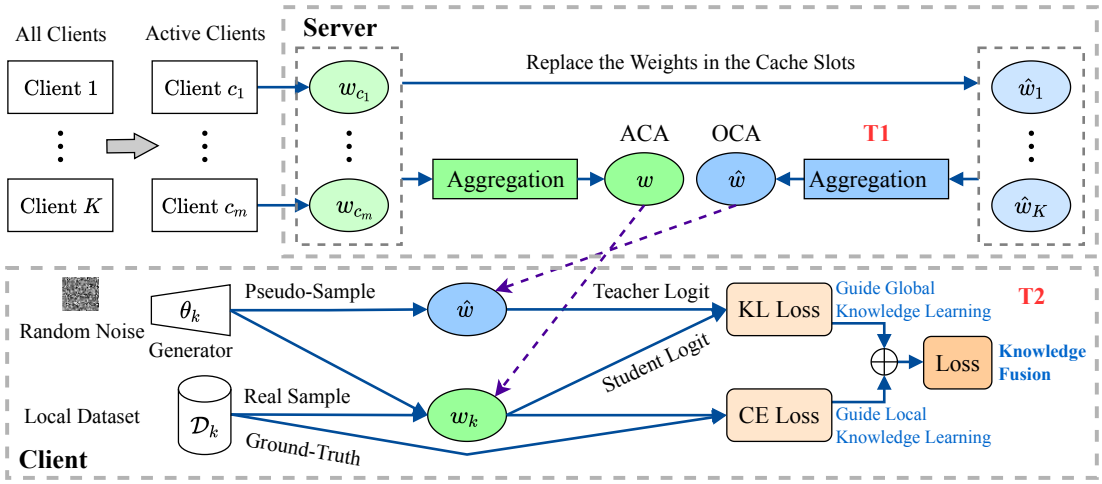


Fig. 1. FedKF Overview. Two key techniques (i.e., T1 and T2) are developed in FedKF. In T1, we develop an active-inactive model aggregation technique to generate an OCA model that represents the global knowledge precisely. In T2, we develop the global-local knowledge fusion technique to enable the local model to learn both the global knowledge (embedded in the teacher model, i.e., the OCA model) and the local knowledge (embedded in the local dataset).

- *Step 5*: Based on the received active clients' local models, the server updates the weights in the corresponding cache slots. The inactive clients' cache slots remain the same. Then, the server re-computes the ACA model and the OCA model. Next, if the model is well trained, then terminate the training process; otherwise, go to Step 1.

When the training is finished, either the ACA model or the OCA model serves as the final model to be used. The detailed FedKF training algorithm is shown in Algorithm 1.

Note that FedKF does not use the server-side generator training approach because the server-side generator training approach achieves much worse model performance (i.e., AMP) than our client-side generator training approach. The main reason is that for the client-side generator training approach, the generator is trained along with the KD process (i.e., the generator training and KD are trained synchronously), so the generator can generate more diversified samples used in KD. In contrast, in the server-side approach, a stationary generator (i.e., well-trained) is used to generate the samples used in KD, so the generated samples are less diversified, resulting in worse model performance.

Two Key Techniques. In FedKF, two key techniques are developed.

- **T1** (active-inactive model aggregation technique): We develop an active-inactive model aggregation technique to generate an OCA model that represents the global knowledge precisely.
- **T2** (global-local knowledge fusion technique): We develop the global-local knowledge fusion technique to enable the local model to learn both the global knowledge (embedded in the teacher model) and the local knowledge (embedded in the local dataset).

For most previous solutions (e.g., FedAvg), only active clients' model weights are aggregated to generate the global model in each round. In contrast, in T1, both active clients' model weights and inactive clients' cached model weights are aggregated to represent the global knowledge. Hence, T1 supports a more precise global knowledge representation. Note that the authors in [28] discuss a solution that uses the global weight to represent the weight of inactive clients in aggregation; it is a coarse global knowledge representation method. T1 is a simple yet precise approach to generating the global model. It is orthogonal to many previous solutions (e.g., FedAvg, FedProx), so T1 can also be used in these solutions to improve their performance. In the following sections, T2 is elaborated.

B. Data-Free Knowledge Distillation

Knowledge distillation (KD) technique enables a student model to learn from one or multiple teacher models [23], [24]. In FedKF, to distill knowledge from the global model (teacher model) to a local model (student model), the data used to train the global model is usually required. However, in FL, data exchange is prohibited due to security concerns (see Definition 4). Accordingly, FedKF employs the idea of data-free KD [25]–[27] to eliminate the requirement of the proxy data on the client side.

In data-free KD, a generator can be trained and then used for generating the imitated training samples. The imitated training samples can be used to transfer knowledge from the teacher model to the student model. Note that the generated imitated training samples do not need to be distributed very similarly to the real training samples. The only requirement is that the generator training samples can be used to facilitate knowledge transfer. Hence, the requirement for the generator in data-free KD and the generator in a traditional GAN is different.

C. Loss Functions for Training Local Generator

Inspired by [26], we design the following loss functions used in the local generator training. To facilitate the description, we first describe the following parameters. Let $g(\theta_k;\cdot)$ be the output of the k-th client's local generator parameterized by θ_k. Let $f(\hat{w};\cdot)$ and $h(\hat{w};\cdot)$ be the feature vector output and the probability vector output of the teacher model parameterized by $\hat{w}$, respectively. On input a random noise

vector $z \sim \mathcal{N}(0, I)$, the generator outputs pseudo-sample $\hat{x}$ with $\hat{x} = g(\theta_k; z)$. On input $\hat{x}$, the teacher model can output probability vector $\hat{\mathbf{p}}$ with $\hat{\mathbf{p}} = h(\hat{w}; \hat{x})$.

Based on the above definitions, the loss functions for training the local generator are introduced as follows.

One-Hot Loss Function. The pseudo-sample $\hat{x}$ is expected to be classified into one particular category concerned by the teacher model with a higher probability. Then, pseudo-label $\hat{y}$ is calculated by $\hat{y} = \arg\max \hat{\mathbf{p}}$. The one-hot loss function is defined as

$$\mathcal{L}_{OH} = \mathbb{E}_{z\sim\mathcal{N}(0,I)} \left[\mathrm{CE} \left(h \left(\hat{w}; g \left(\theta_k; z \right) \right), \hat{y} \right) \right], \tag{2}$$

where CE is cross entropy. If $\mathcal{L}_{OH}$ is minimized, then a generated sample can be classified into one specific class with a significantly high probability. This phenomenon occurs when real samples are used for training.

Information Entropy Loss Function. In order to force the generator to generate samples covering all classes, the information entropy loss is used to measure the uniformity of the class distribution. Specifically, given a probability vector $\mathbf{p} = (p_1, p_2, \ldots, p_d)$, the information entropy of $\mathbf{p}$ is calculated by $\mathrm{IE}(\mathbf{p}) = -\sum_{i=1}^{d} p_i \log p_i$. The information entropy loss can be defined as

$$\mathcal{L}_{IE} = -\mathrm{IE}(\mathbb{E}_{z\sim\mathcal{N}(0,I)} \left[h \left(\hat{w}; g \left(\theta_k; z \right) \right) \right]). \tag{3}$$

When $\mathcal{L}_{IE}$ moves to the minimum, the generator tends to generate samples for each class with roughly the same probability. Thus, minimizing the information entropy loss can result in a training sample set in which the number of samples for each class is roughly the same.

Activation Loss Function. It is observed that the real training sample's feature vector tends to receive a higher activation value. Thus, the activation loss function is defined as

$$\mathcal{L}_{A} = -\mathbb{E}_{z\sim\mathcal{N}(0,I)} [\| f \left(\hat{w}; g \left(\theta_k; z \right) \right) \|_1], \tag{4}$$

where $\|\cdot\|_1$ is the l_1 norm.

Total Loss Function. By taking the above three loss functions into consideration, the total loss function for the generator training is given by

$$\mathcal{L}_{G} = \mathcal{L}_{IE} + \lambda_1 \mathcal{L}_{OH} + \lambda_2 \mathcal{L}_{A}, \tag{5}$$

where λ_1 and λ_2 are hyper parameters for balancing the three loss functions.

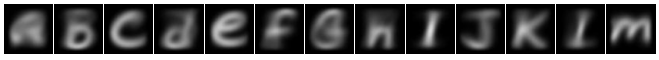
D. Performance of Trained Generator

In KD, theoretically, we can use randomly generated training samples to train the student model to mimic the behavior of the teacher model. Note that a randomly generated training sample can be fed to the teacher model to get its label. Then, the properly labeled samples can be used for the student model training. However, this approach has low efficiency and usually cannot achieve high KD accuracy. In FedKF, for the generator used in KD, if the generator can generate samples that are distributed relatively close to the real-world training samples, the KD process can be finished with high accuracy. The generated samples do not need to be as accurate as some other applications (e.g., the generator required by deepfake [29]).

In what follows, we conduct some experiments to demonstrate the performance of the trained generator used in data-free KD. In the experiments, we train a teacher model on centralized real-world training data. Then, we use the teacher model to train a generator with the loss function (as defined in Eq. 5) in a data-free manner. Meanwhile, we use the generated samples to train a student model via KD.

The experiment settings are briefly introduced as follows. The optimizers used in training the teacher model, student model, and generator are SGD, SGD, and Adam, respectively. As for training the teacher and student model, the learning rate is set to 0.01 for LeNet-5 [30] and 0.1 for ResNet-8 [31]. For training the generator, the learning rate is set to 0.2, 0.02, and 0.02 on the EMNIST [32], CIFAR-10, and CIFAR-100 [32] dataset, respectively. For hyper parameters λ_1 and λ_2 in training the generator, we set them to $\{0.2, 0.02\}$ on the EMNIST dataset, $\{0.01, 0.002\}$ on the CIFAR-10 dataset, and $\{0.01, 0.002\}$ on the CIFAR-100 dataset.

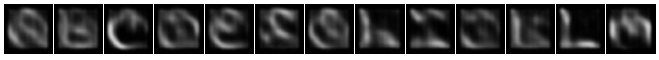
Fig. 2 and 3 show the visualization results of averaged images on the EMNIST dataset and the generated dataset (using the local generator), respectively. Although the generated images are not very similar to the real images used in training, but it is sufficient for achieving good performance in KD (as demonstrated by the experimental results below).



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m)

(n) (o) (p) (q) (r) (s) (t) (u) (v) (w) (x) (y) (z)

Fig. 2. Visualization of the averaged image in each category (from a to z) on the EMNIST dataset.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m)

(n) (o) (p) (q) (r) (s) (t) (u) (v) (w) (x) (y) (z)

Fig. 3. Visualization of the averaged image in each category (from a to z) on the generated dataset (using the local generator).

Table IV reports the performance of the teacher and student models on the different datasets. The teacher models achieve 93.35%, 90.78%, and 67.14% accuracies on the EMNIST, CIFAR-10, and CIFAR-100 datasets, respectively. The student models using T2 obtain 92.25%, 89.05%, and 63.45% accuracies without any real-world training data. The performance of the student model is just slightly lower than the teacher model.

TABLE IV
PERFORMANCE OF THE TEACHER AND STUDENT MODELS IN T2 ON THE DIFFERENT DATASETS.

Datasets	Teacher		Student	
	Model	Accuracy (%)	Model	Accuracy (%)
EMNIST	LeNet-5	93.35	LeNet-5	92.25
CIFAR-10	ResNet-8	90.78	ResNet-8	89.05
CIFAR-100	ResNet-8	67.14	ResNet-8	63.45

E. Loss Functions for Training Local Model

The local model is trained by T2 (global-local knowledge fusion technique). The loss functions are introduced as follows.

KL Loss Function. FedKF allows each client to use the imitated training samples generated by the generator to distill the global knowledge from the teacher model to the local model. Meanwhile, the local model learns the local knowledge from the local dataset.

Let $h(w_k;\cdot)$ be the probability vector output of the k-th client's local model parameterized by w_k. We define the knowledge distillation loss as

$$\mathcal{L}_{KL} = \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\mathrm{KL}\left(h(\hat{w};g(\theta_k;z)) \parallel h(w_k;g(\theta_k;z))\right)\right], \quad (6)$$

where KL stands for Kullback–Leibler divergence [33]. When minimizing $\mathcal{L}_{KL}$, the local model is moving closer to the teacher model (i.e., learning the global knowledge).

Cross Entropy Loss Function. We define the loss function over the local dataset $\mathcal{D}_k$ as

$$\mathcal{L}_{CE} = \mathbb{E}_{(x,y)\sim\mathcal{D}_k}\left[\mathrm{CE}\left(h(w_k;x),y\right)\right]. \quad (7)$$

When minimizing $\mathcal{L}_{CE}$, the local model is learning the local knowledge (embedded in the local dataset).

Total Loss Function. The total loss function for global-local knowledge fusion is given as

$$\mathcal{L} = \mathcal{L}_{CE} + \gamma\mathcal{L}_{KL}, \quad (8)$$

where γ is a hyperparameter for balancing the two loss functions. When minimizing $\mathcal{L}$, the global-local knowledge is fused to the local model.

F. FedKF Training Algorithm

The detailed FedKF training algorithm is shown in Algorithm 1. The FedKF training algorithm requires K (the number of all clients), C (selection rate), T (the number of communication rounds), E (the number of local training epochs), B (local batch size), β (the learning rate for training local generator), η (the learning rate for training local model), θ (the initial generator weight) and w^0 (the initial ACA model weight) as inputs and returns w^T (the final ACA model) and $\hat{w}^T$ (the final OCA model) as outputs. In line 2, the server initializes the OCA model $\hat{w}^0$ and all the models $\{\hat{w}_k\}_{k=1}^K$ in the cache slots with w^0 and all clients initialize their local generators $\{\theta_k\}_{k=1}^K$ with θ. In line 4, the server uniformly selects m clients as active ones with $m = C\cdot K$ at random. In line 6, each active client executes **ClientUpdate** and uploads the latest local model to the server. In line 7, the server replaces models in the cache slots with the latest models uploaded from active clients in the current round. In line 9, the server aggregates all the latest models uploaded from active clients in the current round and gets the updated ACA model. In line 10, the server aggregates all models in the cache slots and gets the updated OCA model.

Algorithm 1: FedKF Training.

Input: K, C, T, E, B, β, η, θ, w^0.
Output: w^T, $\hat{w}^T$.

1 **Server executes:**
2 $\hat{w}^0 \leftarrow w^0$, $\hat{w}_k \leftarrow w^0$, $\theta_k \leftarrow \theta$ $(k = 1,\ldots,K)$.
3 **for** *round* $t = 1, 2, \ldots, T$ **do**
4 $\mathcal{S}_t \leftarrow$ (random set of $m = C\cdot K$ active clients).
5 **for** *each client* $k \in \mathcal{S}_t$ ***in parallel*** **do**
6 $w_k^t \leftarrow$ **ClientUpdate**$(k, w^{t-1}, \hat{w}^{t-1})$.
7 $\hat{w}_k \leftarrow w_k^t$. // Update the model in the k-th cache slot.
8 **end**
9 $w^t \leftarrow \frac{1}{\sum_{k\in\mathcal{S}_t}|\mathcal{D}_k|}\sum_{k\in\mathcal{S}_t}|\mathcal{D}_k| w_k^t$.
10 $\hat{w}^t \leftarrow \frac{1}{\sum_{k=1}^K|\mathcal{D}_k|}\sum_{k=1}^K|\mathcal{D}_k|\hat{w}_k$.
11 **end**
12
13 **ClientUpdate** $(k, w, \hat{w})$**:**
14 $w_k \leftarrow w$.
15 $\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size B).
16 **for** *each local epoch* $i = 1, 2, \ldots, E$ **do**
17 **for** *each batch* $b \in \mathcal{B}$ **do**
18 $\theta_k \leftarrow \theta_k - \beta\cdot\nabla\mathcal{L}_G(\theta_k)$. // Update generator via minimizing $\mathcal{L}_G$.
19 $w_k \leftarrow w_k - \eta\cdot\nabla\mathcal{L}(w_k)$. // Update local model via minimizing $\mathcal{L}$.
20 **end**
21 **end**
22 **return** w_k.

V. FEDKF ANALYSIS

In this section, we analyze FedKF from four aspects: why high AMP, why high fairness, why privacy-preserving, and its relationship with agnostic FL.

A. Why High Average Model Performance

There are two techniques contributing to the high AMP of FedKF.

T1 Helps to Improve AMP. On the server side, most previous solutions use the active clients aggregated (ACA) model as the global model that is used for inference. In contrast, our solution FedKF (OCA) uses the overall clients aggregated (OCA) model as the global model. Since the ACA model only aggregates a small portion of clients, it may lead to a large AMP degradation when clients' local datasets are non-IID. In contrast, when T1 is used to generate the global model on the server side, it aggregates a model that contains more precise global knowledge learned during FL training. Thus, T1 can significantly increase the AMP of FedKF.

T2 Helps to improve AMP. In FedAvg, the AMP degradation is caused by the client model drift issue when training on heterogeneous data. To improve AMP, FedKF uses T2 to address the client model drift issue. On the client side, when performing the local training, each client learns the global knowledge simultaneously. T2 can regularize the local training by jointly considering both the global and local knowledge. It can avoid the local model overfitting towards the local dataset. Thus, the client model drift issue is alleviated, and the AMP of FedKF is boosted.

B. Why High Fairness

There are two techniques contributing to the high fairness of FedKF.

T1 Helps to Improve Fairness. On the server side, most previous solutions use the ACA model as the global model, while our solution FedKF (OCA) uses the OCA model as the global model. Because the ACA model only aggregates a small portion of clients, it may generate a model that is biased towards only the active clients. Thus, the ACA model has poor fairness when data is heterogeneous. On the contrary, if T1 is used to generate the global model, both the inactive and the active clients are taken into consideration, leading to a fairer model.

T2 Helps to Improve Fairness. In FedAvg, the local model is trained only on the local dataset, so the local model could be overfitted on the local dataset. It leads to different degrees of overfitting on different clients when their local datasets are non-IID. Hence, the AMP variance could be very large, and the model fairness could be low. In contrast, T2 can be used to avoid the local model overfitting towards the local dataset since both global knowledge (embedded in the teacher model) and local knowledge (embedded in the local dataset) are fused into the local model. Therefore, T2 can help FedKF to achieve higher model fairness.

C. Why Privacy-Preserving

In each FedKF training round, there are two information flows exchanged between the server and each client. First, the server needs to send two models (i.e., the ACA and OCA models) to each client. Second, each client needs to send the updated local model to the server after the global-local knowledge fusion. Hence, no information about the local data is shared directly. According to Definition 4, FedKF is privacy-preserving.

D. Relationship with Agnostic FL

The traditional FL is to optimize the model on the global distribution. In practice, the target distribution can be very different from the global distribution. To improve the applicability of FL, agnostic federated learning (AFL) is proposed [34]. AFL aims to optimize the model performance on any possible target distribution formed by a mixture of client distributions. In other words, AFL has better domain generalization capability. Therefore, it captures more use cases and significantly expands the applicability of FL.

The mathematical description of AFL is presented as follows. Let Dis_k denote the local data distribution of k-th client. The global distribution U is denoted as $U = \sum_{k=1}^{K} \frac{n_k}{n} Dis_k$, where n_k represents the number of k-th client's local samples and $n = \sum_{k=1}^{K} n_k$. In AFL, the target distribution $\widehat{U}$ can be modeled as an unknown mixture of the distributions $\{Dis_1, Dis_2, \ldots, Dis_K\}$. That is, $\widehat{U} = \sum_{k=1}^{K} \widehat{p}_k Dis_k$, where $\widehat{p}_k \geq 0$ and $\sum_{k=1}^{K} \widehat{p}_k = 1$. AFL aims to optimize the model performance on $\widehat{U}$ for any possible choices of $\widehat{p}_k$ $(k = 1, \ldots, K)$.

For a model trained by AFL, it can be used for many different agnostic target domains. Each agnostic target domain represents a distinct use case. A good model in AFL is expected to have high AMP and high fairness across these multiple use cases in reality. Thus, a good AFL model should be able to achieve both high AMP and high fairness in heterogeneous AFL. In the following, we theoretically prove that a model trained by FedKF can directly have both high AMP and high fairness in heterogeneous AFL.

***Lemma** 1:* We denote by w a trained model via using FedKF. In heterogeneous FL with FedKF, let $\Omega = \{Dis_1, \ldots, Dis_K\}$ represent a set of the client distributions. WLP_w^{Ω} denotes the worst-case local performance of w on Ω. Suppose that w is used for an arbitrary agnostic domain $\widehat{U}$, let $MP_w^{\widehat{U}}$ be the model performance on the agnostic domain $\widehat{U}$. It holds that

$$MP_w^{\widehat{U}} \geq WLP_w^{\Omega}. \tag{9}$$

Proof 5.1: Let a_w^k represent the test accuracy on distribution Dis_k $(k = 1, \ldots, K)$. Then, WLP_w^{Ω} is given by

$$WLP_w^{\Omega} = \min\{a_w^1, \ldots, a_w^K\}. \tag{10}$$

For $MP_w^{\widehat{U}}$, we have

$$MP_w^{\widehat{U}} = \sum_{k=1}^{K} \widehat{p}_k a_w^k. \tag{11}$$

According to Eq. (10), WLP_w^{Ω} is the lower bound for a_w^k $(k = 1, \ldots, K)$. Thus, substituting WLP_w^{Ω} for a_w^k $(k = 1, \ldots, K)$ in Eq. (11), it holds that

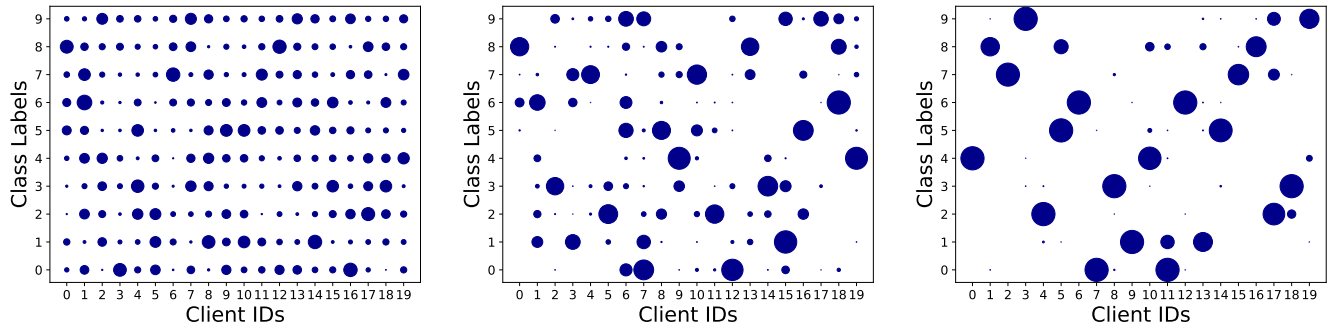
$$MP_w^{\widehat{U}} \geq \sum_{k=1}^{K} \widehat{p}_k WLP_w^{\Omega} = WLP_w^{\Omega} \sum_{k=1}^{K} \widehat{p}_k = WLP_w^{\Omega}.$$

Theorem 5.1: Suppose that there are multiple (e.g., Q) arbitrary agnostic target domains $\widehat{U}_i$ $(i = 1, \ldots, Q)$. Let $WLP_w^{\widehat{\Omega}}$ be the worst-case performance on these agnostic target domains, where $\widehat{\Omega} = \{\widehat{U}_1, \ldots, \widehat{U}_Q\}$. It holds that

$$WLP_w^{\widehat{\Omega}} \geq WLP_w^{\Omega}, \tag{12}$$

Proof 5.2: According to Lemma 1, it holds that $MP_w^{\widehat{U}_i} \geq WLP_w^{\Omega}$ for any $i \in [Q]$. Since $WLP_w^{\widehat{\Omega}} = \min\{MP_w^{\widehat{U}_i}, \ldots, MP_w^{\widehat{U}_M}\}$, we have $WLP_w^{\widehat{\Omega}} \geq WLP_w^{\Omega}$.

According to Theorem 5.1, the WLP of a model trained by FedKF in heterogeneous FL is the lower bound of the WLP when the model is used for heterogeneous AFL. Given the fact that the WLP metric tends to measure the joint performance of AMP and FM, FedKF directly turns out to be a good solution



(a) $\alpha = 1$ (b) $\alpha = 0.1$ (c) $\alpha = 0.01$

Fig. 4. Visualization of statistical heterogeneity among clients on CIFAR-10 dataset with different α. The size of scattered points is proportional to the number of training samples for a label available on the client.

to achieve high AMP and high fairness simultaneously in heterogeneous AFL. Since AFL has more use cases, FedKF can also be applied to a broader range of use cases. Thus, FedKF has much broader impacts in reality.

VI. Experiments

This section first introduces the experiment setup. Then, the experimental results are reported.

A. Experiment Setup

FedKF Variants. When FedKF training is finished, either the ACA model or the OCA model can serve as the final model to be used. Depending on which model is used as the final model, FedKF has two variants: FedKF (ACA) and FedKF (OCA). Besides, during the local training in FedKF, the OCA model serves as the teacher model, and the ACA model serves as the local model, so in each communication round, the amount of data in downlink communication (from server to client) is doubled compared with FedAvg. If we use the ACA model to serve as both the teacher model and the local model, then the amount of data in downlink communication does not increase in each communication round. This FedKF variant is denoted as FedKF-. We call it the communication-efficient variant in this paper. Note that in FedKF-, the student model is trained based on both the KL loss and the CE loss, so the student model (starting from the ACA model) is evolving along with the local training process. In contrast, the teacher model (always the ACA model) is fixed during the local training. To sum up, we have four FedKF variants: FedKF (ACA), FedKF (OCA), FedKF- (ACA), and FedKF- (OCA).

Solutions in Comparison. We compare four FedKF variants with previous FL algorithms, including FedAvg [1], FedProx [10], FedGen [12], FedGKD [13], and q-FFL [14]. For q-FFL, we use FedAvg as its optimization method, and it is also called q-FedAvg in [14].

Datasets. We conduct experiments on three datasets, including EMNIST [32], CIFAR-10, and CIFAR-100 [35]. For EMNIST, we only use a subset of the dataset by randomly sampling 10% from each class. Each client's local dataset is split into 80% training set and 20% testing set randomly. Following previous works [11]–[13], [16], we use Dirichlet distribution to model heterogeneous data. The Dirichlet distribution $\mathbf{Dir}_K(\alpha)$ has a adjustable concentration parameter α. A smaller α implies a higher data heterogeneity across different clients. For example, the statistical heterogeneity among clients on CIFAR-10 with different concentration parameters α is shown in Fig. 4.

Implementation & Training Details. The proposed FedKF and solutions in comparison are all implemented in PyTorch [36] and evaluated on a Linux server with two TITAN RTX GPUs. Following most of the previous studies on FL, we simulate the computation of IoT devices (i.e., clients) on the Linux server and then measure FedKF performance. Since the learning process is exactly the same, the performance metrics measured are accurate in our experiments.

For the shared global model, two different neural network models are used. ResNet-8 [31] is used for CIFAR-10/100 and LeNet-5 [30] is used for EMNIST. Note that Batch Normalization (BN) fails on heterogeneous training data due to the statistics of running mean and variance for the clients' data [37]; we replace BN with Group Normalization (GN) to produce stabler results and set the number of channels of each group as 1. For the generator used in data-free KD, we use a deep convolutional generator used in [38] and replace $\tanh$ activation function in the last layer by $\mathrm{sigmoid}$.

The optimizers used in training local generator and local model are Adam and SGD, respectively. For local generator training, the learning rate is set to 0.001. For local model training, the learning rate is set to 0.01 and 0.1 for LeNet-5 and ResNet-8, respectively. For FL learning, we run a total of 100 communication rounds. We set the number of all clients K to 20 and the selection rate C to 20%, which means there are 4 clients selected as active ones in each round. The number of local update epochs E is set to 10, and local batch size B is set to 64. For FedKF, we set γ, λ_1, and λ_2 to 1, 0.1, and 0.1, respectively, for all the datasets.

To compare with FedProx, we tune FedProx's parameter μ from $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$ and report the best result. For FedProx, the best μ for CIFAR-10, CIFAR-100, and EMNIST are 0.001, 0.0001, and 0.001, respectively. To compare with FedGKD, following [13], we set the default buffer size as 5. For FedGKD, we tune FedGKD's parameter γ from $\{0.001, 0.01, 0.1, 0.2, 0.5, 1\}$. The best γ for CIFAR-10, CIFAR-100, and EMNIST are 0.2, 0.2, and 0.001, respectively.

TABLE V
PERFORMANCE OF DIFFERENT SOLUTIONS ON EMNIST.

Solutions	$\alpha = 1$			$\alpha = 0.1$			$\alpha = 0.01$		
	AMP (%)	FM ($\times 10^{-3}$)	WLP (%)	AMP (%)	FM ($\times 10^{-3}$)	WLP (%)	AMP (%)	FM ($\times 10^{-2}$)	WLP (%)
FedAvg	83.78 ± 0.24	1.477 ± 0.132	76.46 ± 1.66	75.61 ± 0.92	7.972 ± 0.703	57.07 ± 1.03	55.42 ± 2.78	4.871 ± 0.976	5.61 ± 2.86
FedProx	83.87 ± 0.18	2.008 ± 0.167	75.24 ± 1.32	75.59 ± 0.99	7.844 ± 0.610	56.75 ± 4.09	55.70 ± 1.82	6.535 ± 1.324	3.96 ± 1.96
FedGen	84.81 ± 0.36	1.379 ± 0.206	78.25 ± 2.29	77.37 ± 1.62	5.405 ± 1.559	61.20 ± 6.59	56.07 ± 3.94	5.951 ± 1.194	9.08 ± 3.92
FedGKD	83.63 ± 0.28	1.463 ± 0.124	74.78 ± 1.34	75.93 ± 0.67	6.477 ± 0.504	56.02 ± 2.44	57.86 ± 1.48	4.266 ± 0.846	7.92 ± 2.32
q-FFL	84.07 ± 0.34	1.729 ± 0.182	76.58 ± 1.78	75.67 ± 0.02	6.740 ± 1.524	57.25 ± 2.30	54.90 ± 2.36	3.128 ± 0.784	18.70 ± 4.36
FedKF- (ACA)	85.18 ± 0.38	$\mathbf{1.064 \pm 0.109}$	$\mathbf{78.38 \pm 0.90}$	82.82 ± 0.50	2.753 ± 0.503	72.71 ± 1.09	74.44 ± 0.95	1.667 ± 0.137	39.92 ± 2.27
FedKF- (OCA)	85.26 ± 0.42	1.211 ± 0.217	78.35 ± 0.95	$\mathbf{83.36 \pm 0.62}$	$\mathbf{2.358 \pm 0.287}$	$\mathbf{73.76 \pm 0.80}$	$\mathbf{76.20 \pm 0.78}$	$\mathbf{1.283 \pm 0.157}$	$\mathbf{41.91 \pm 4.68}$
FedKF (ACA)	$\mathbf{85.54 \pm 0.37}$	1.717 ± 0.251	77.39 ± 1.02	82.76 ± 0.16	3.049 ± 0.848	72.41 ± 0.12	72.33 ± 1.12	2.153 ± 0.478	38.12 ± 3.42
FedKF (OCA)	85.46 ± 0.29	1.540 ± 0.192	78.26 ± 0.88	82.94 ± 0.12	2.978 ± 0.383	71.53 ± 1.57	72.73 ± 1.09	2.780 ± 0.574	36.73 ± 3.86

TABLE VI
PERFORMANCE OF DIFFERENT SOLUTIONS ON CIFAR-10.

Solutions	$\alpha = 1$			$\alpha = 0.1$			$\alpha = 0.01$		
	AMP (%)	FM ($\times 10^{-3}$)	WLP (%)	AMP (%)	FM ($\times 10^{-2}$)	WLP (%)	AMP (%)	FM ($\times 10^{-2}$)	WLP (%)
FedAvg	74.28 ± 0.28	1.014 ± 0.148	68.23 ± 0.43	61.89 ± 0.81	1.807 ± 0.240	34.41 ± 7.74	38.48 ± 0.43	7.914 ± 0.612	1.82 ± 1.21
FedProx	74.00 ± 0.16	1.079 ± 0.126	67.87 ± 0.23	62.25 ± 0.77	1.764 ± 0.365	36.09 ± 10.66	38.30 ± 0.64	7.979 ± 0.742	1.24 ± 0.63
FedGen	74.52 ± 0.22	0.778 ± 0.110	68.56 ± 0.06	62.92 ± 0.75	1.916 ± 0.342	34.00 ± 6.26	40.34 ± 0.51	6.673 ± 0.689	2.63 ± 2.41
FedGKD	74.83 ± 0.18	0.786 ± 0.096	69.40 ± 0.38	63.98 ± 0.38	1.508 ± 0.682	40.04 ± 5.72	39.14 ± 0.72	5.478 ± 0.574	2.40 ± 1.76
q-FFL	73.96 ± 0.31	$\mathbf{0.716 \pm 0.102}$	68.59 ± 0.19	61.49 ± 1.26	1.853 ± 0.451	33.02 ± 8.20	37.75 ± 0.58	3.655 ± 0.484	1.26 ± 0.78
FedKF - (ACA)	75.19 ± 0.07	0.737 ± 0.158	$\mathbf{70.14 \pm 0.49}$	67.97 ± 0.79	1.333 ± 0.075	47.72 ± 3.30	47.98 ± 0.97	5.576 ± 0.347	3.96 ± 3.67
FedKF - (OCA)	$\mathbf{75.62 \pm 0.13}$	1.150 ± 0.138	69.13 ± 0.59	69.88 ± 0.29	0.874 ± 0.187	53.89 ± 2.54	54.41 ± 0.54	3.063 ± 0.191	26.09 ± 4.84
FedKF (ACA)	75.23 ± 0.17	0.971 ± 0.117	69.68 ± 0.42	67.86 ± 0.39	1.271 ± 0.146	45.12 ± 2.75	47.89 ± 0.49	5.817 ± 0.536	5.68 ± 1.28
FedKF (OCA)	75.54 ± 0.09	1.197 ± 0.078	69.03 ± 0.27	$\mathbf{70.11 \pm 0.74}$	$\mathbf{0.835 \pm 0.110}$	$\mathbf{55.18 \pm 1.32}$	$\mathbf{54.74 \pm 0.66}$	$\mathbf{2.792 \pm 0.263}$	$\mathbf{29.86 \pm 5.51}$

TABLE VII
PERFORMANCE OF DIFFERENT SOLUTIONS ON CIFAR-100.

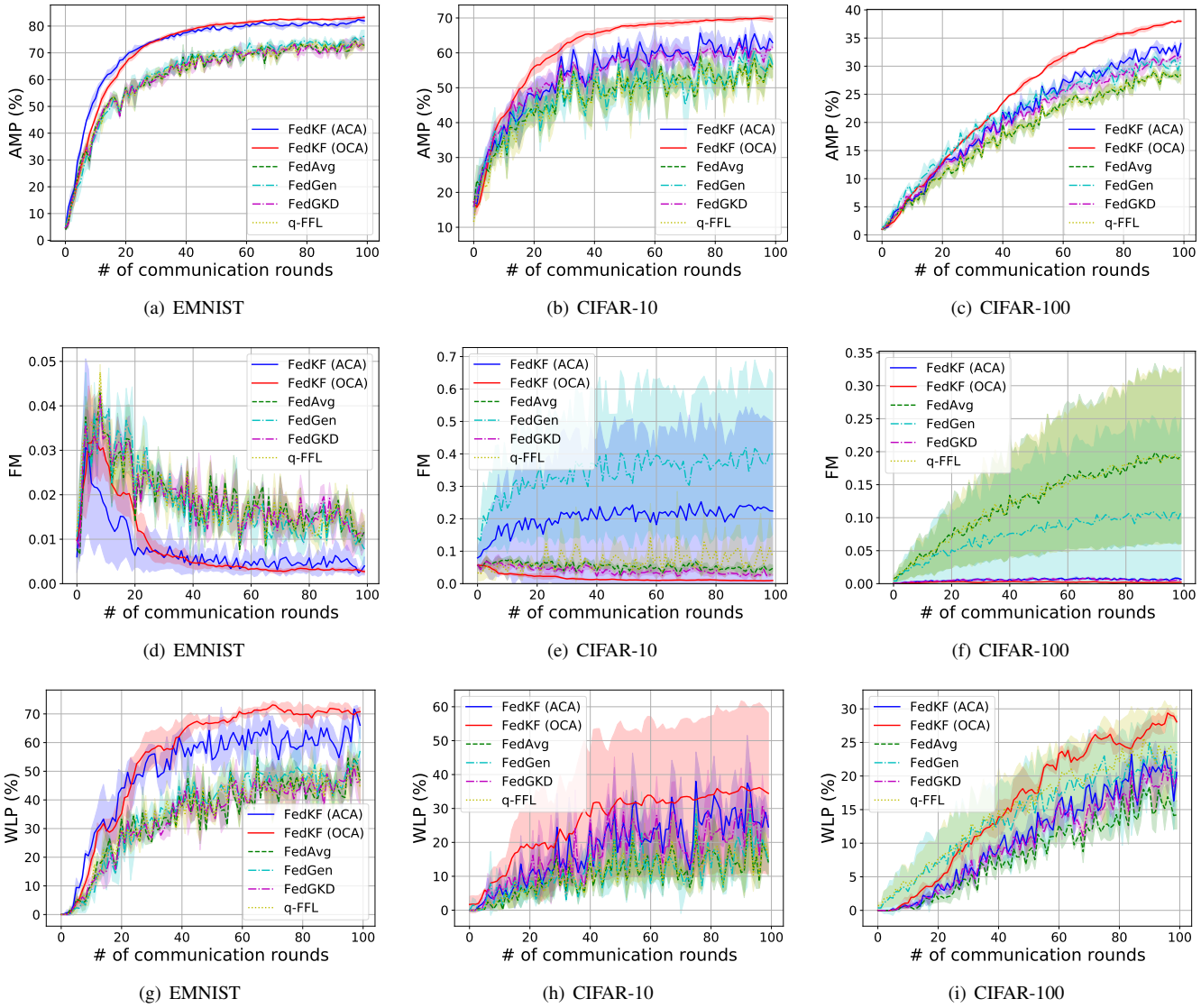
Solutions	$\alpha = 1$			$\alpha = 0.1$			$\alpha = 0.01$		
	AMP (%)	FM ($\times 10^{-3}$)	WLP (%)	AMP (%)	FM ($\times 10^{-3}$)	WLP (%)	AMP (%)	FM ($\times 10^{-2}$)	WLP (%)
FedAvg	36.92 ± 0.42	1.549 ± 0.274	29.53 ± 0.62	29.37 ± 0.58	6.265 ± 1.206	17.70 ± 0.64	17.41 ± 0.12	1.660 ± 0.152	3.37 ± 0.33
FedProx	36.63 ± 0.49	0.958 ± 0.241	30.60 ± 0.74	29.62 ± 0.51	6.131 ± 0.455	17.11 ± 1.11	17.75 ± 0.19	1.828 ± 0.134	3.41 ± 0.43
FedGen	40.23 ± 0.38	1.154 ± 0.222	32.21 ± 0.80	32.08 ± 0.51	5.378 ± 0.938	19.37 ± 4.49	18.06 ± 0.05	1.727 ± 0.122	1.93 ± 0.39
FedGKD	38.45 ± 0.56	1.302 ± 0.213	32.97 ± 0.58	32.23 ± 0.46	3.960 ± 1.218	22.51 ± 1.02	15.77 ± 0.18	1.626 ± 0.146	1.06 ± 0.34
q-FFL	36.12 ± 0.31	$\mathbf{0.729 \pm 0.178}$	31.26 ± 0.67	29.43 ± 0.96	5.781 ± 2.946	17.59 ± 2.74	16.82 ± 0.22	1.757 ± 0.118	1.06 ± 0.19
FedKF - (ACA)	39.86 ± 0.49	0.952 ± 0.113	33.79 ± 0.58	33.73 ± 0.73	4.420 ± 1.189	24.50 ± 1.21	20.30 ± 0.58	1.404 ± 0.134	3.34 ± 0.77
FedKF - (OCA)	41.13 ± 0.41	1.019 ± 0.329	35.51 ± 1.08	37.43 ± 0.65	1.869 ± 0.192	28.30 ± 1.21	23.45 ± 0.66	0.999 ± 0.198	8.03 ± 0.69
FedKF (ACA)	40.53 ± 0.37	0.906 ± 0.152	34.16 ± 0.72	34.11 ± 0.63	4.273 ± 0.709	22.30 ± 1.31	21.32 ± 0.78	1.900 ± 0.055	4.50 ± 0.16
FedKF (OCA)	$\mathbf{41.30 \pm 0.43}$	0.978 ± 0.137	$\mathbf{36.43 \pm 0.88}$	$\mathbf{38.08 \pm 0.19}$	$\mathbf{1.775 \pm 0.109}$	$\mathbf{29.40 \pm 0.70}$	$\mathbf{24.06 \pm 0.50}$	$\mathbf{0.876 \pm 0.089}$	$\mathbf{8.71 \pm 0.97}$

To compare with q-FFL, we tune q-FFL's parameter q from $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$. The best q for CIFAR-10, CIFAR-100, and EMNIST are 0.0001, 0.0001, and 0.0001, respectively.

B. Experimental Results

Performance Metrics. Table V-VII show the performance metrics (i.e., AMP, FM, and WLP) of different solutions for three degrees of data heterogeneity ($\alpha = 1, 0.1, 0.01$). In the tables, for each case, the best result is marked as **bold** font, and the second best is marked using underline. We have the following six findings.

- **F1:** FedKF-(OCA) performance is the best when $\alpha = 0.1$ and 0.01 on EMNIST, while FedKF (OCA) performance is the best when $\alpha = 0.1$ and 0.01 on both CIFRA-10 and CIFAR-100 datasets.
- **F2:** When $\alpha = 1$ on three datasets, at least one of the four FedKF variants can rank in the top 2 (in terms of AMP, FM, and WLP) among all solutions in comparison.
- **F3:** FedKF has better performance than FedGen on all three datasets. FedGen allows the leakage of local label counts to the server, meaning that the server knows exactly how each local dataset is heterogeneous. Thus, it violates the privacy-preserving property (see Definition 4). FedKF outperforms FedGen while still guaranteeing the privacy-preserving property.
- **F4:** Under most cases, the OCA variants have better performance than their corresponding ACA variants. The



(a) EMNIST (b) CIFAR-10 (c) CIFAR-100

(d) EMNIST (e) CIFAR-10 (f) CIFAR-100

(g) EMNIST (h) CIFAR-10 (i) CIFAR-100

Fig. 5. Performance v.s. number of communication rounds on different datasets with $\alpha = 0.1$.

superiority of the OCA variants (over the ACA variants) is more obvious with the decrease of α (i.e., higher data heterogeneity).

- **F5:** Under most cases, the original FedKF has a better performance than its communication-efficient variants (i.e., FedKF-(ACA) and FedKF-(OCA)).
- **F6:** Under most cases, the superiority of FedKF variants (over other solutions) is more obvious with a decreasing of α. For example, on CIFAR-10, FedKF (OCA)'s WLP is (69.03%-68.23%)=0.8% better than FedAvg when $\alpha = 1$, whereas it is (29.86%-1.82%)=28.04% better than FedAvg when $\alpha = 0.01$. Hence, FedKF is especially good at dealing with highly heterogeneous data.

AMP v.s. Number of Rounds. As shown in Fig. 5, FedKF has a faster learning speed (i.e., has higher communication efficiency) than prior solutions when data is heterogeneous (i.e., $\alpha = 0.1$). Specifically, the number of communication rounds to achieve the same AMP as running FedAvg for 100 rounds on different datasets with $\alpha = 0.1$ is shown in Table VIII. We can observe that the number of communication rounds is significantly reduced by using FedKF. For example,

TABLE VIII
THE NUMBER OF ROUNDS NEEDED IN DIFFERENT SOLUTIONS TO ACHIEVE THE SAME AMP AS RUNNING FEDAVG FOR 100 ROUNDS ON DIFFERENT DATASETS WITH $\alpha = 0.1$. THE SPEEDUP OF A SOLUTION IS COMPUTED AGAINST FEDAVG.

Solutions	EMNIST		CIFAR-10		CIFAR-100	
	# of rounds	Speedup	# of rounds	Speedup	# of rounds	Speedup
FedAvg	100	1×	100	1×	100	1×
FedProx	>100	<1×	98	1×	96	1×
FedGen	77	1.3×	91	1.1×	74	1.4×
FedGKD	>100	<1×	49	2.0×	82	1.2×
q-FFL	100	1×	>100	<1×	100	1×
FedKF- (ACA)	**31**	**3.2**×	35	2.9×	68	1.5×
FedKF- (OCA)	**31**	**3.2**×	30	3.3×	55	1.8×
FedKF (ACA)	**31**	**3.2**×	34	2.9×	66	1.5×
FedKF (OCA)	32	3.1×	**28**	**3.6**×	**52**	**1.9**×

FedKF (OCA) needs 32, 28, and 52 communication rounds to achieve the same AMP as running FedAvg for 100 rounds on EMNIST, CIRFA-10, and CIFAR-100, respectively. Thus, FedKF is much more communication-efficient than prior solutions.

FM v.s. Number of Rounds. The FM v.s. number of communication rounds on different datasets with $\alpha = 0.1$ is shown in Fig. 5. We find that the FMs of FedKF (OCA) are always the lowest among all solutions on all datasets when training is finished, which means FedKF (OCA) achieves better fairness compared with previous solutions. Besides, the FM curves of FedKF (OCA) are smoother and have smaller fluctuation amplitude compared with other solutions on all the datasets. Furthermore, compared with FedKF (ACA), FedKF (OCA) has lower FMs, smoother FM curves, and smaller fluctuation amplitude, which means the technique T1 indeed improves model fairness in FedKF.

WLP v.s. Number of Rounds. The WLP v.s. number of communication rounds on different datasets with $\alpha = 0.1$ is shown in Fig. 5. When training is finished, all the WLPs of our solution FedKF (OCA) are highest among all solutions on all datasets. On EMNIST and CIFAR-10, the WLPs of both FedKF (ACA) and FedKF (OCA) are higher than previous solutions, while FedKF (OCA) has more smooth WLP curves compared with FedKF (ACA).

Communication Data Amount. For both FedKF (ACA) and FedKF (OCA), in each communication round, their uplink communication (from client to server) data amount is the same as FedAvg, while their downlink communication data amount is doubled compared with FedAvg. For both FedKF- (ACA) and FedKF- (OCA), their communication data amount is the same as FedAvg. Therefore, compared with FedKF, FedKF- slightly sacrifices performance to ensure there is no communication traffic increase in each communication round. Besides, as shown in Table VIII, FedKF and FedKF- require much fewer communication rounds in learning compared with FedAvg. Hence, in most cases, FedKF and FedKF- require less communication data amount than prior solutions. Note that FedKF- has the least communication consumption since it requires half of the downlink communication data amount in each communication round compared with FedKF.

Client's Computation Overhead. We have a theoretical analysis of the client's computation overhead as follows. In the local training phase of federated learning, the main computation is forward propagation (FP) and backpropagation (BP) through the models. To compare the computation overhead of different solutions in the local training phase, we compare the numbers of the FP and BP operations through the models for each batch of the local data. Table IX demonstrates the numbers of the FP and BP operations through the models for each batch of the local data in the local training phase in different solutions. In FedGen, only the last layers of the local model (i.e., predictor) need 2 operations of both FP and BP for each batch of the local data, and the number of the FP and BP operations in the remaining layers is the same as FedAvg. Therefore, we consider the numbers of the FP and BP operations through the classifier in FedGen as 1.5 and 1.5, respectively. In FedGKD, the teacher model (the classifier) needs to output the logit of each sample for knowledge distillation, so the number of the FP operations through the classifier is 2. In FedKF, the numbers of the FP and BP operations through the teacher model (the classifier) are 1 and 1, respectively, and those through the student model (the classifier) are 2 and 2.

TABLE IX
THE NUMBERS OF THE FP AND BP OPERATIONS THROUGH THE MODELS FOR EACH BATCH OF THE LOCAL DATA IN THE LOCAL TRAINING PHASE IN DIFFERENT SOLUTIONS.

Solutions	Classifier		Generator	
	Forward	Backward	Forward	Backward
FedAvg	1	1	0	0
FedProx	1	1	0	0
FedGen	1.5	1.5	1	0
FedGKD	2	1	0	0
q-FFL	1	1	0	0
FedKF-	3	3	1	1
FedKF	3	3	1	1

Time Consumption. To intuitively compare the time complexity of different solutions, we conduct experiments to report the time consumption of different solutions for 100 communication rounds on different datasets with $\alpha = 0.1$. Note that the active clients in the experiments serially execute **ClientUpdate** one by one, which means the time consumption in real-world applications is much less than that in the experiments due to its parallel execution of **ClientUpdate**. As shown in Table X, while FedKF- and FedKF require every client independently to train an extra local generator and conduct local knowledge distillation, the time consumption of our solutions is about $1.8\times$-$1.9\times$ longer than FedAvg on all three different datasets, which is acceptable. Our solutions take more time to achieve better capabilities to handle data heterogeneity in FL.

TABLE X
TIME CONSUMPTION OF DIFFERENT SOLUTIONS FOR 100 COMMUNICATION ROUNDS ON DIFFERENT DATASETS WITH $\alpha = 0.1$. THE SPEED-DOWN COMPUTES ITS TIME CONSUMPTION RATIO COMPARED TO FEDAVG.

Solutions	EMNIST		CIFAR-10		CIFAR-100	
	Time	Speed-down	Time	Speed-down	Time	Speed-down
FedAvg	6min 43s	$1\times$	45min 12s	$1\times$	45min 27s	$1\times$
FedProx	7min 20s	$1.1\times$	47min 39s	$1.1\times$	49min 12s	$1.1\times$
FedGen	8min 11s	$1.2\times$	50min 23s	$1.1\times$	50min 31s	$1.1\times$
FedGKD	7min 5s	$1.1\times$	47min 43s	$1.1\times$	48min 11s	$1.1\times$
q-FFL	7min 4s	$1.1\times$	47min 53s	$1.1\times$	48min 3s	$1.1\times$
FedKF-	12min 3s	$1.8\times$	84min 36s	$1.9\times$	84min 40s	$1.9\times$
FedKF	12min 4s	$1.8\times$	84min 38s	$1.9\times$	84min 42s	$1.9\times$

Table XI reports the time consumption in different solutions to achieve the same AMP as running FedAvg for 100 rounds on different datasets with $\alpha = 0.1$. In FL, the total time consumption is obtained by (the training time per round)$\times$(number of rounds needed to converge). As shown in Table XI, to achieve the same level of model performance, our solutions are generally faster than others on the EMINST and CIFAR-10 datasets and slower on the CIFAR-100 dataset.

TABLE XI
TIME CONSUMPTION NEEDED IN DIFFERENT SOLUTIONS TO ACHIEVE THE SAME AMP AS RUNNING FEDAVG FOR 100 ROUNDS ON DIFFERENT DATASETS WITH $\alpha = 0.1$. THE SPEEDUP OF A SOLUTION IS COMPUTED AGAINST FEDAVG.

Solutions	EMNIST		CIFAR-10		CIFAR-100	
	Time	Speedup	Time	Speedup	Time	Speedup
FedAvg	6min 43s	1×	45min 12s	1×	45min 27s	1×
FedProx	>6min 43s	<1×	>45min 12s	<1×	>45min 27s	<1×
FedGen	6min 18s	1.1×	>45min 12s	<1×	**37min 23s**	**1.2**×
FedGKD	>6min 43s	<1×	**23min 23s**	**1.9**×	39min 31s	1.2×
q-FFL	>6min 43s	<1×	>45min 12s	<1×	>45min 27s	<1×
FedKF- (ACA)	**3min 44s**	**1.8**×	29min 37s	1.5×	>45min 27s	<1×
FedKF- (OCA)	**3min 44s**	**1.8**×	25min 23s	1.8×	>45min 27s	<1×
FedKF (ACA)	**3min 44s**	**1.8**×	28min 47s	1.6×	>45min 27s	<1×
FedKF (OCA)	3min 52s	1.7×	23min 42s	1.9×	44min 3s	1×

Thus, the computation overhead of FedKF and its variants are comparable to other solutions.

Robustness. It can be found from Fig. 5 that FedKF is more robust than other solutions (in terms of performance stability) during FL training. The fluctuation amplitude of AMP in FedKF (OCA) is extremely small compared with FedAvg, FedGen, FedGKD, and q-FFL when data is heterogeneous.

TABLE XII
PERFORMANCE COMPARISON BETWEEN FEDAVG AND FEDAVG (OCA) ON DIFFERENT DATASETS WITH $\alpha = 0.1$.

Datasets	Metrics	FedAvg	FedAvg (OCA)
EMNIST	AMP (%)	75.61 ± 0.92	$78.38 \pm 0.64 \uparrow$
	FM ($\times 10^{-3}$)	7.972 ± 0.703	$5.329 \pm 1.378 \downarrow$
	WLP (%)	57.07 ± 1.03	$60.88 \pm 1.87 \uparrow$
CIFAR-10	AMP (%)	61.89 ± 0.81	$65.82 \pm 0.22 \uparrow$
	FM ($\times 10^{-2}$)	1.807 ± 0.240	$1.194 \pm 0.142 \downarrow$
	WLP (%)	34.41 ± 7.74	$44.21 \pm 4.58 \uparrow$
CIFAR-100	AMP (%)	29.37 ± 0.58	$34.76 \pm 0.22 \uparrow$
	FM ($\times 10^{-3}$)	6.265 ± 1.206	$1.853 \pm 0.304 \downarrow$
	WLP (%)	17.70 ± 0.64	$25.78 \pm 0.92 \uparrow$

Using T1 to Improve FedAvg. As mentioned before, we can use T1 to improve prior solutions. Let FedAvg (OCA) represent the solution using T1. It simply uses the OCA model as the final model to be used. Note that, during FedAvg (OCA) training, the global model broadcasted to active clients is still the ACA model. Table XII and Fig. 6 show the performance comparison between FedAvg and FedAvg (OCA) on different datasets with $\alpha = 0.1$. In Table XII, it can be found that all metrics of FedAvg (OCA) are better than FedAvg. Fig. 6 demonstrates that FedAvg (OCA) has a faster learning speed and smaller fluctuation amplitude than FedAvg.

To further explore under what circumstances T1 can improve FedAvg, we study the impacts of α (the concentration parameter of the Dirichlet distribution) and K (the number of overall clients) on the performance of the global models (i.e., the ACA model and the OCA model). Fig. 7 shows the AMP of FedAvg v.s. α w/ and w/o T1 on different datasets with different K, where "O" stands for the OCA model, "A"

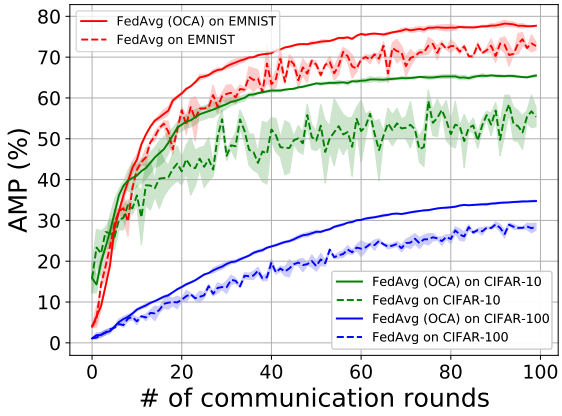


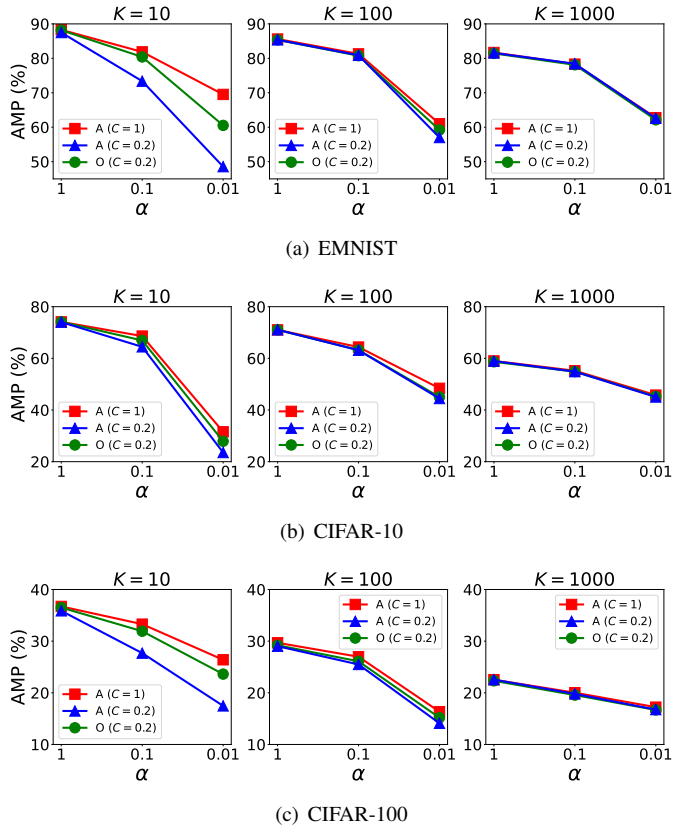
Fig. 6. AMP of FedAvg w/ and w/o T1 on different datasets with $\alpha = 0.1$.

stands for the ACA model, and C represents the selection rate of active clients. Note that $C = 1$ represents full device participation that is unrealistic in most scenarios due to the presence of stragglers and the increase in communication cost per communication round (e.g., the communication cost per communication round of $C = 1$ is 5 times as large as that of $C = 0.2$). In Fig. 7, it can be found that the AMP of $C = 1$ is always the highest, which is consistent with the theoretical proof in [28]. When the number of overall clients K is limited, the more heterogeneous the data, the larger the gap in the AMP of the ACA model between $C = 0.2$ and $C = 1$. In this case, compared to the ACA model, the OCA model (that additionally aggregates the latest historical local models of inactive clients) can obtain higher AMP and narrow the gap.

Fig. 8 shows the AMP of FedAvg v.s. selection rate w/ and w/o T1 on different datasets with $\alpha = 0.1$, where the ACA model is the aggregation of only active clients, and the OCA model is the aggregation of both active and inactive clients by using T1. Note that when the selection rate is 5%, only one client is selected as active in each round. In Fig. 8, it is easy to find that the performance (i.e., AMP) of the OCA model is always higher than that of the ACA model when the selection rate is less than 1. Besides, the ACA model's performance degrades fast with the descent of the selection rate, i.e., by about 13% on the EMNIST dataset, about 20% on the CIFAR-10 dataset, and about 14% on the CIFAR-100 dataset, which means the ACA model's performance is vulnerable to the low selection rate on non-IID data. Fortunately, with T1, the OCA model's performance degrades slightly from 80% to 5% of the selection rate, especially on the CIFAR-100 dataset. Therefore, the improvement using T1 (in terms of global model performance) is more obvious with a decreasing selection rate.

VII. CONCLUSION

In this paper, we have developed FedKF to handle data heterogeneity for IoT devices in FL. Two novel techniques are developed to achieve the precise global knowledge representation and global-local knowledge fusion, by which the local model drift issue can be alleviated. We theoretically prove that



(a) EMNIST

(b) CIFAR-10

(c) CIFAR-100

Fig. 7. AMP of FedAvg v.s. α w/ and w/o T1 on different datasets with different K.

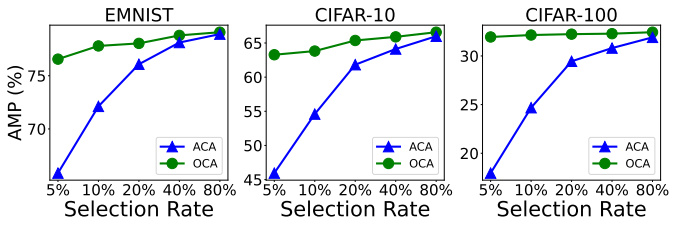


Fig. 8. AMP of FedAvg v.s. selection rate w/ and w/o T1 on different datasets with $\alpha = 0.1$.

FedKF can directly turn out to be a good solution in heterogeneous agnostic FL, so FedKF has much broader application scenarios. According to theoretical analysis and experimental results, FedKF achieves the three design goals (i.e., high model performance, high model fairness, and privacy-preserving) simultaneously. In summary, the proposed techniques can mitigate data heterogeneity issues and significantly boost FL performance for IoT devices. There are two directions to launch future work. First, we plan to test FedKF on more datasets and make the testbeds more diverse. Second, we plan to improve FedKF further and make it more lightweight in terms of communication and computation overhead.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[2] J. Zheng, K. Li, N. Mhaisen, W. Ni, E. Tovar, and M. Guizani, "Exploring deep reinforcement learning-assisted federated learning for online resource allocation in privacy-preserving edgeiot," *IEEE Internet of Things Journal (IoT-J)*, 2022.

[3] D. Wu, R. Ullah, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "Fedadapt: Adaptive offloading for iot devices in federated learning," *IEEE Internet of Things Journal (IoT-J)*, 2022.

[4] T. T. Vu, D. T. Ngo, H. Q. Ngo, M. N. Dao, N. H. Tran, and R. H. Middleton, "Joint resource allocation to minimize execution time of federated learning in cell-free massive mimo," *IEEE Internet of Things Journal (IoT-J)*, 2022.

[5] N. Al-Maslamani, B. S. Ciftler, M. Abdallah, and M. M. Mahmoud, "Towards secure federated learning for iot using drl-enabled reputation mechanism," *IEEE Internet of Things Journal (IoT-J)*, 2022.

[6] Y. Liu, Y. Dong, H. Wang, H. Jiang, and Q. Xu, "Distributed fog computing and federated learning enabled secure aggregation for iot devices," *IEEE Internet of Things Journal (IoT-J)*, 2022.

[7] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning (ICML)*, 2020, pp. 5132–5143.

[8] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in Neural Information Processing Systems (NIPS)*, pp. 7611–7623, 2020.

[9] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data *et al.*, "A field guide to federated optimization," *arXiv preprint arXiv:2107.06917*, 2021.

[10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems (MLSys)*, pp. 429–450, 2020.

[11] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *arXiv preprint arXiv:2006.07242*, 2020.

[12] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *International Conference on Machine Learning (ICML)*, 2021, pp. 12 878–12 889.

[13] D. Yao, W. Pan, Y. Dai, Y. Wan, X. Ding, H. Jin, Z. Xu, and L. Sun, "Local-global knowledge distillation in heterogeneous federated learning with non-iid data," *arXiv preprint arXiv:2107.00051*, 2021.

[14] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[15] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," *Advances in Neural Information Processing Systems (NIPS)*, pp. 5972–5984, 2021.

[16] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 713–10 722.

[17] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, "Federated learning meets multi-objective optimization," *IEEE Transactions on Network Science and Engineering (TNSE)*, vol. 9, no. 4, pp. 2039–2051, 2022.

[18] C. T Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems (NIPS)*, pp. 21 394–21 405, 2020.

[19] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems (NIPS)*, pp. 3557–3568, 2020.

[20] M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *arXiv preprint arXiv:1901.09888*, 2019.

[21] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022.

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

[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

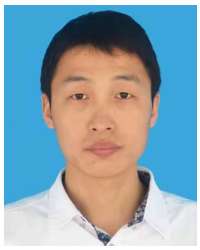
[24] T. Fukuda, M. Suzuki, G. Kurata, S. Thomas, J. Cui, and B. Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers." in *Interspeech*, 2017, pp. 3697–3701.

[25] R. G. Lopes, S. Fenu, and T. Starner, "Data-free knowledge distillation for deep neural networks," *arXiv preprint arXiv:1710.07535*, 2017.

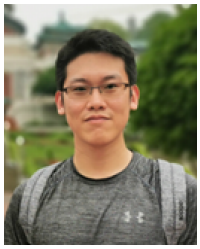
[26] H. Chen, Y. Wang, C. Xu, Z. Yang, C. Liu, B. Shi, C. Xu, C. Xu, and Q. Tian, "Data-free learning of student networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2019, pp. 3514–3522.
[27] G. Fang, J. Song, C. Shen, X. Wang, D. Chen, and M. Song, "Data-free adversarial distillation," *arXiv preprint arXiv:1912.11006*, 2019.
[28] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations (ICLR)*, 2019.
[29] M. Westerlund, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9, no. 11, 2019.
[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, pp. 541–551, 1989.
[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
[32] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2921–2926.
[33] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, pp. 79–86, 1951.
[34] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning (ICML)*, 2019, pp. 4615–4625.
[35] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Technical Report, University of Toronto*, 2009.
[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and e. a. Antiga, Luca, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 8026–8037.
[37] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 4387–4398.
[38] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

Xu Zhou received his Master's degree with Computer Technology from Zhengzhou University, Zhengzhou, Henan, China, in 2023. Before that, he received his Bachelor's degree with Energy Chemical Engineering from Changzhou University, Changzhou, Jiangsu, China. He is currently pursuing the Ph.D. degree in Computer Science. His research interests are federated learning and AI security.

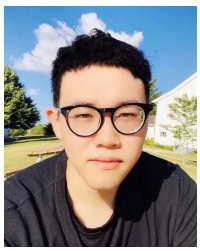
Xinyu Lei (Member, IEEE) received the Ph.D. degree with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, in 2021. He is currently an Assistant Professor with the Department of Computer Science, Michigan Technological University, Houghton, MI, USA. He worked as a Research Assistant with the Texas A&M University at Qatar, Doha, Qatar, in 2013. In 2017, he worked as a Research Intern with Ford Motor Company, Dearborn, MI, USA. His current research focuses on machine learning and cybersecurity.



Cong Yang is currently a lecturer at the School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou. He received his Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, in 2017, and his B.S. degree in information security from Chongqing University, Chongqing, in 2010. His current research interests include deep neural networks, computer vision, and medical image processing.



Yichun Shi received his PhD degree from Michigan State University, East Lansing, MI, USA, in 2021. His research focuses on machine learning and computer vision. He is currently a research scientist at ByteDance, CA, USA. He worked as a research intern at NEC Lab of America, San Jose, CA, USA. He worked as a research intern at Visa, Foster City, CA, USA.



Xiao Zhang received his Ph.D. degree in Computer Science at Michigan State University, USA, in 2023. His research interests are in the areas of next-generation wireless networking (e.g., optical wireless communication (OWC) and mmWave), AIoT, HCI and mobile computing. He is currently a Postdoctoral Associate at Duke University, USA.

Jingwen Shi is currently a Ph.D. student in the Department of Computer Science and Engineering at Michigan State University. She received her Bachelor's degree from Hunan University in 2016 and her Master's degree from the University of Chinese Academy of Sciences in 2019. Her research interests include cellular network security, Android security, AI for security, and cloud computing.