Efficient Federated Learning for AIoT Applications Using Knowledge Distillation

Tian Liu, Jun Xia, *Graduate Student Member, IEEE*, Zhiwei Ling, Xin Fu, *Senior Member, IEEE*, Shui Yu, *Senior Member, IEEE*, and Mingsong Chen, *Senior Member, IEEE*

***Abstract*—As a promising distributed machine learning paradigm, federated learning (FL) trains a central model with decentralized data without compromising user privacy, which makes it widely used by Artificial Intelligence Internet of Things (AIoT) applications. However, the traditional FL suffers from model inaccuracy, since it trains local models only using hard labels of data while useful information of incorrect predictions with small probabilities is ignored. Although various solutions try to tackle the bottleneck of the traditional FL, most of them introduce significant communication overhead, making the deployment of large-scale AIoT devices a great challenge. To address the above problem, this article presents a novel distillation-based FL (DFL) method that enables efficient and accurate FL for AIoT applications. By using knowledge distillation (KD), in each round of FL training, our approach uploads both the soft targets and local model gradients to the cloud server for aggregation, where the aggregation results are then dispatched to AIoT devices for the next round of local training. During the DFL local training, in addition to hard labels, the model predictions approximate soft targets, which can improve model accuracy by leveraging the knowledge of soft targets. To further improve our DFL model performance, we design a dynamic adjustment strategy of loss function weights for tuning the ratio of KD and FL, which can maximize the synergy between soft targets and hard labels. Comprehensive experimental results on well-known benchmarks show that our approach can significantly improve the model accuracy of FL without introducing significant communication overhead.**

***Index Terms*—Artificial Intelligence Internet of Things (AIoT), dynamic adjustment strategy, federated learning (FL), knowledge distillation (KD), model accuracy.**

Manuscript received 26 May 2022; revised 15 November 2022; accepted 9 December 2022. Date of publication 15 December 2022; date of current version 7 April 2023. This work was supported in part by the Natural Science Foundation of China under Grant 62272170; in part by the National Key Research and Development Program of China under Grant 2018YFB2101300; in part by the Shanghai Trusted Industry Internet Software Collaborative Innovation Center; and in part by the "Digital Silk Road" Shanghai International Joint Laboratory of Trustworthy Intelligent Software under Grant 22510750100. *(Tian Liu and Jun Xia are co-first authors.) (Corresponding author: Mingsong Chen.)*

Tian Liu is with the MoE Engineering Research Center of Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200062, China, and also with the Department of Information Science and Engineering, Zaozhuang University, Zaozhuang 277160, China (e-mail: liutian2534@qq.com).

Jun Xia, Zhiwei Ling, and Mingsong Chen are with the MoE Engineering Research Center of Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200062, China (e-mail: jxia@stu.ecnu.edu.cn; 51215902044@stu.ecnu.edu.cn; mschen@sei.ecnu.edu.cn).

Xin Fu is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA (e-mail: xfu8@central.uh.edu).

Shui Yu is with the School of Computer Science, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: shui.yu@uts.edu.au).

Digital Object Identifier 10.1109/JIOT.2022.3229374

I. Introduction

ALONG with the proliferation of artificial intelligence (AI) and Internet of Things (IoT), federated learning (FL) [1], [2], [3], [4], [5] techniques are increasingly used in safety-critical AI IoT (AIoT) applications (e.g., autonomous driving, commercial surveillance, and industrial control [6], [7]). As shown in Fig. 1, different from centralized machine learning, FL enables keeping data samples distributed while sharing the sample knowledge among all the AIoT devices. In FL, the cloud server is responsible for dispatching and aggregating model gradients rather than collecting samples from AIoT devices through the network, which can greatly reduce the communication overhead and protect the data privacy of AIoT devices during the model training process.

Although FL enables effective collaboration among AIoT devices and the cloud server, it drastically suffers from its model inaccuracy caused by the loss of knowledge during model training [8]. The optimization objective of FL local training is to minimize the distance between the correct prediction and the hard label and ignore all the incorrect predictions [1]. However, the ignoring of incorrect predictions results in the loss of knowledge since the knowledge is a learned mapping from input vectors to output vectors, and all the sample-to-prediction mappings are part of the knowledge according to [8] and [9]. The probability of incorrect predictions represents the similarities between the current sample and other different categories. Therefore, the traditional FL based on hard labels loses some knowledge during the model training process, resulting in decreased FL model accuracy.

Since knowledge distillation (KD) can enhance the model knowledge and the model generalization ability, it is used to improve the model accuracy [8]. During the "student model" training process, there are two optimization objectives, i.e., hard labels of data and soft targets from the "teacher model." The loss function of the "student model" is defined as the sum of the cross-entropy loss function (i.e., the distance between model predictions and the corresponding hard labels of data) and the Kullback–Leibler divergence loss function (i.e., the distance between model predictions and the corresponding soft targets from the "teacher model"). As an online paradigm of KD, federated distillation (FD) implements collaborative training of

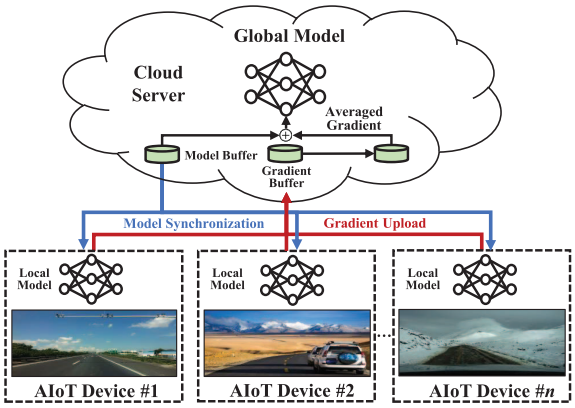

Fig. 1. Common architecture of FL.

different device models only by interacting soft targets between the cloud server and all the devices [10], [11], [12]. However, all these methods focus more on the fundamental problems of network resource limitation for large-scale architecture rather than the FL performance improvement.

To improve the FL model accuracy, various methods have been proposed, e.g., global control variable-based methods [13], [14], reinforcement learning-based methods [15], device grouping-based methods [16], [17], and KD-based methods [18]. However, all these mentioned methods improve FL performance using complex reinforcement learning strategies or global variables with large sizes. Therefore, most of them are unsuitable for AIoT applications with limited network and memory resources. Moreover, these KD-based methods require collecting data distribution and sample categories from all devices or constructing public data sets, which brings huge risks to data privacy protection. Therefore, how to design an efficient and accurate FL without introducing significant communication overhead and ensuring data privacy is becoming a great challenge in AIoT design.

In order to address the above challenges, this article presents a novel distillation-based FL (DFL) method named DFL that can effectively enhance the model knowledge during the FL training process. Unlike the traditional FL that only trains models based on hard labels of device samples, our proposed DFL method set two optimization objectives for the model, i.e., the hard labels of data samples and the corresponding soft targets. We aggregate label-wise sample logits as the soft targets of the "teacher model" and dispatch them together with the global model for FL model training, which introduces negligible extra network overhead as the soft targets size is always much smaller than the global model. In this way, our DFL method can increase the model accuracy by incorporating the knowledge of soft targets into the model training. This article makes the following three major contributions.

1) To improve the model accuracy of DFL, we present a novel architecture that combines the merits of both global soft targets and model gradients for the purpose of knowledge enhancement.
2) To wisely utilize the knowledge represented by soft targets, we design a dynamic adjustment strategy, which can tune the ratio of loss functions of soft targets and hard labels during the DFL training.
3) We conduct both theoretical and empirical analysis on the convergence of DFL and prove that DFL converges as fast as FedAvg in arbitrarily heterogeneous data scenarios.

We implement our approach using our proposed DFL architecture and the dynamic adjustment strategy. Comprehensive experimental results show that our proposed approach can achieve better performance than state-of-the-art methods without introducing drastic communication overhead.

The remainder of this article is organized as follows. After the introduction to related works in Sections II and IV gives the details of our DFL approach. Section V presents the experimental results, showing the effectiveness of our approach. Finally, Section VI concludes this article.

II. Related Work

As more and more safety-critical AIoT applications adopt FL, the FL model accuracy is becoming a major concern in AIoT design. In order to improve the FL model inference accuracy, various methods have been investigated. For example, Karimireddy et al. [13] proposed a method named SCAFFOLD, using global control variables to correct the "client-drift" in the local training process. Similar to SCAFFOLD, Huang et al. [14] presented a method employing the federated attentive message passing to promote more cooperation among similar devices. However, all these two methods upload/dispatch additional large-size controllers (i.e., the global control variables and the attentive messages) together with model gradients between the cloud server and devices. By using built-in generators, Zhu et al. [21] proposed a data-free KD approach named FedGen to address the problem of heterogeneous FL. Lin et al. [18] proposed an ensemble distillation method that trains the central model with unlabeled data and the corresponding outputs of device models. Nonetheless, these two methods cannot be directly used in real scenarios, since they require each device to upload their data distribution or sample categories, which brings the risks of data exposure and huge communication overhead. Moreover, it is impractical to construct built-in generators or public data sets, though they are helpful for model training.

Due to the merits of model knowledge sharing and enhancement, various online versions of KD (i.e., FD) have been investigated in AIoT applications. For example, Anil et al. [9] proposed the co-distillation method with data samples shared by all the AIoT devices. Based on FD and federated data augmentation, Jeong et al. [11] used generative adversarial networks (GANs) [19] to generate a public data set and carried out KD on the public data set during the model training process. By leveraging an unlabeled public data set, Itahara et al. [20] proposed a distillation-based semi-supervised FL algorithm that exchanges outputs of local models among mobile devices. However, all these FD approaches above focus on reducing communication overhead rather than improving model accuracy. Moreover, these methods with

public data sets introduce inevitable risks of privacy exposure, which cannot be ignored in practice.

Although KD techniques are promising in enhancing the model performance, their combination with FL in AIoT scenarios still suffers from the limited inference improvements coupled with significant communication overhead. To the best of our knowledge, so far existing KD-based approaches do not consider that the knowledge of soft targets changes along with the model training process. Our work is the first attempt that fully explores the synergy between the model gradients and global soft targets to further enable knowledge sharing among AIoT devices. Due to the enhanced knowledge obtained by soft targets using our proposed architecture and dynamic adjustment strategy, the accuracy of DFL models can be significantly improved, while the communication overhead is negligible.

III. Preliminary Knowledge

A. Federated Learning

As a promising distributed machine learning technology, FL [1] is proposed to solve the data silos problem, where all the involved clients can share their knowledge without exposing local data privacy. Assume that there are a total of K devices in an AIoT system, and in the rth FL training round there are N ($N \leq K$) devices being selected to dispatch the global model. Let w and w^k be the models on the cloud server and the kth client, respectively. Let the number of samples on the kth device be l_k, where the sum of l_k of the selected N devices is l, and the loss function of device models is Φ. The objective of FL is defined as follows:

$$\min_{w} \; F(w) \triangleq \sum_{k=1}^{N} \frac{l_k}{l} \Phi(w). \tag{1}$$

In the rth training round, let w_r be the global model dispatched to all the selected devices for the purpose of local training. At the end of the rth training round, the local model of the kth selected device will be updated as follows:

$$w^k \leftarrow w_r + \eta \Delta_r^k \tag{2}$$

where η is the learning rate and Δ_r^k is the model gradient (i.e., weight differences) achieved by the local training of device k in round r. Meanwhile, to improve communication efficiency, all the selected AIoT devices need to upload their model gradients (i.e., Δ) rather than the models to the cloud server. After receiving all the local models of selected devices, the cloud server will conduct the aggregation operation to obtain a new global model as follows:

$$w_{r+1} = \; w_r + \frac{\sum_{k=1}^{K} l_k \times \Delta_r^k}{\sum_{k=1}^{K} l_k}. \tag{3}$$

B. Knowledge Distillation

To improve the inference performance of a model (i.e., student model), KD resorts to the advanced knowledge learned by a well-trained model (i.e., teacher model) and distills it to the student model. Typically, the knowledge learned by the student model comes from two sources: 1) soft labels that are the predictions by the teacher model and 2) hard labels that are a set of labeled training data samples provided for the distillation purpose. Suppose that there are two neural network models (i.e., student model s and teacher model t) participating in KD, whose training data and corresponding labels are x and y, respectively. The loss function of the student model is defined as follows:

$$\mathcal{L}(s) = \mathcal{F}(y|s, Y) + \mathcal{G}(y|s, y|t) \tag{4}$$

where $\mathcal{F}(y|s, Y)$ is the cross-entropy loss function between the predictions $y|s$ and hard labels Y. $\mathcal{G}(y|s, y|t)$ is the Kullback–Leibler divergence loss function, which indicates the distance between the predictions of student model $y|s$ and the corresponding teacher predictions $y|t$. Based on this loss function and the provided training data, after the distillation, the inference performance of the student models can be significantly improved.

IV. Our DFL Approach

Typically, an AIoT application involves a cloud server and plenty of AIoT devices, where each AIoT device has limited communication and memory capacities. In this article, we focus on the model performance rather than the problem of incentive or fairness. Therefore, we assume that at the beginning of our DFL architecture deployment, data samples are collected by each device and used for local model training. The model to be trained is initially placed on the cloud server and dispatched to AIoT devices at the beginning of each training round. Since soft targets can enhance the model knowledge and the model generalization ability [8], our approach introduces KD into our architecture to improve the model accuracy. Unlike existing FD methods, our DFL approach uploads/dispatches model gradients and soft targets (generated using local samples in the previous round) simultaneously during the interaction between AIoT devices and the cloud server. When a new device joins the AIoT application, it will receive the latest global model and soft targets from the cloud server and perfectly fit into the DFL model training. The proposed DFL model training procedure is divided into two parts: 1) the cloud server training part, which includes the dispatching, aggregation, and update of both model gradients and soft targets and 2) the local training part, which trains local models using both local samples and dispatched soft targets. Once the local training finishes, an AIoT device needs to figure out new label-wise sample logits for the following aggregation. The following sections will detail the key components and the convergence analysis of our DFL approach.

A. Architecture and Workflow of Our DFL Approach

Fig. 2 depicts the overall architecture and workflow of our proposed DFL approach for some AIoT application, which mainly consists of three parts: 1) the FL processing part (marked in yellow); 2) the soft targets processing part in the cloud server (marked in blue); and 3) the soft targets processing part in AIoT devices (marked in red). Unlike traditional

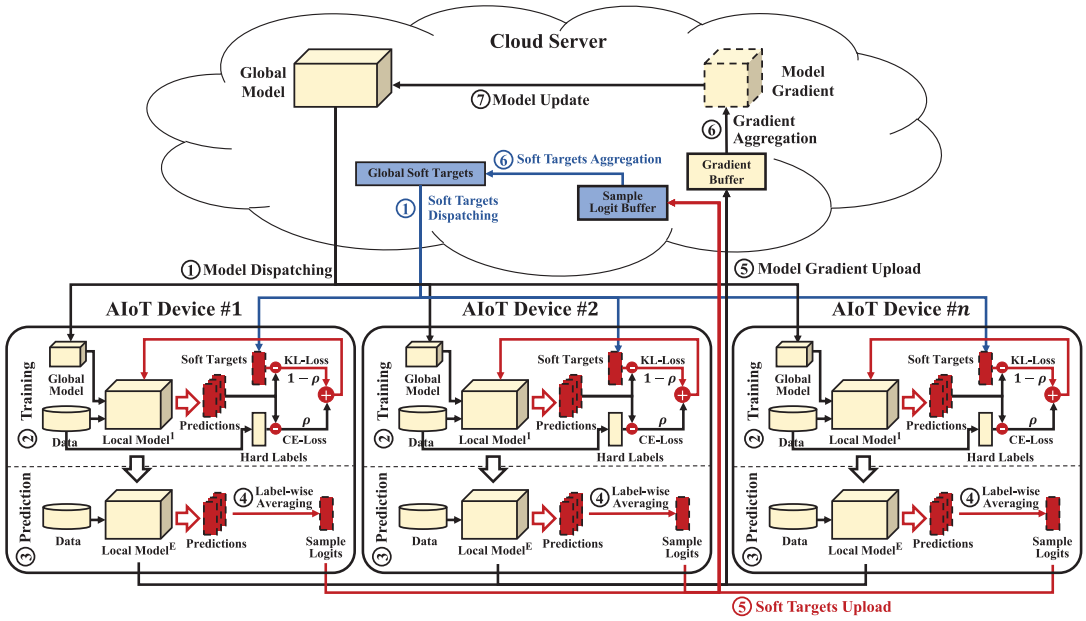

Fig. 2. Architecture and workflow of our DFL approach.

FL methods, in our approach the cloud server needs to aggregate label-wise sample logits collected from devices to form the global soft targets, which will be dispatched to selected AIoT devices for the distillation purpose. Specifically, our DFL method involves following seven steps.

Step 1 (Model and Soft Targets Dispatching): At the beginning of each training round, the cloud server dispatches both the global soft targets and global model to the selected AIoT device (see Algorithm 1 for more details).

Step 2 (Local Training): Once receiving the dispatched global soft targets and global model, each selected device will conduct the local training (i.e., distillation) using both the hard labels of local data and received soft targets [see (7)]. Similar to traditional KD methods, our DFL approach adopts both KL-Loss and CE-Loss for the distillation so that the predictions of local models can be accurately approximated to corresponding labels in each training epoch (see Algorithm 2 for more details).

Step 3 (Prediction): After E epochs of local training, each selected device will use the newly trained local model to generate one prediction (i.e., soft label) for each of its local data samples.

Step 4 (Label-Wise Averaging): Assume that there are a total of N sample categories in the AIoT application. For each category of local data samples in one device, our approach averages their predictions to generate one sample logit. By combining sample logits of all the categories together, we can form the local soft targets for the device, which are in the form of an $N \times N$ matrix.

Step 5 (Soft Targets and Gradient Upload): After the local training, each device needs to upload the newly generated label-wise soft targets (i.e., sample logits) and model gradients to the cloud server for aggregation.

Step 6 (Soft Targets and Gradient Aggregation): Once the cloud server receives both model gradients and soft targets (i.e., averaged sample logits) of all the selected devices, it will average all the collected local gradients and soft targets.

Step 7 (Global Model Update): The cloud server forms a new global model based on the averaged global gradients.

DFL will repeats all the above seven steps continuously until the convergence of global models.

B. Training Procedure of DFL

The model training procedure of our DFL approach consists of two parts, i.e., the cloud server procedure and the local update procedure. At the very beginning of the model training, AIoT devices randomly collect a set of data samples and save them in their local memory for model training, while the cloud server initializes the global model and soft targets. Similar to the classic FL method (i.e., FedAvg [1]), our approach randomly selects a fraction of AIoT devices participating in each round of model training due to the limited network resources of real AIoT applications. The collaboration of the cloud server and AIoT devices of our proposed DFL method will be detailed in the following two sections.

1) Cloud Server Procedure: When the model training of our DFL approach starts, the cloud server first dispatches the current global model and soft targets to the selected AIoT devices. After receiving the latest model and soft targets, the selected AIoT devices will conduct several epochs of local training, respectively. At the end of round r, we upload both model gradients and the newly generated label-wise sample logits of all the selected AIoT devices to the cloud server for aggregations using the following formulas:

$$w_{r+1} = w_r + \frac{\sum_{k=1}^{K} |D_k| \times \Delta_{r+1}^k}{\sum_{k=1}^{K} |D_k|} \tag{5}$$

$$\mathscr{Y}_{r+1} = \frac{\sum_{k=1}^{K} |D_k| \times \mathscr{Y}_{r+1}^k}{\sum_{k=1}^{K} |D_k|} \tag{6}$$

Algorithm 1: Cloud Server Procedure of DFL

Input: i) N, # of total AIoT devices;
ii) c, fraction of devices on each round;
iii) R, # of training rounds;
iv) $D = \{D_1, \ldots, D_N\}$, set of datasets;

1. Initialize($w, \mathscr{Y}$);
2. $K \leftarrow \text{Max}(c \cdot N, 1)$;
for $r \leftarrow 1$ *to* R **do**
  3. $S \leftarrow$ random set of K devices;
  4. Dispatch($w_r, \mathscr{Y}_r, S$);
  for *each device* $k \in S$ **do**
    5. $(\Delta^k_{r+1}, \mathscr{Y}^k_{r+1}) \leftarrow \text{DeviceUpdate}(w_r, \mathscr{Y}_r)$;
  end
  6. $w_{r+1} = w_r + \frac{\sum_{k=1}^{K} |D_{Idx(S_k)}| \times \Delta^k_{r+1}}{\sum_{k=1}^{K} |D_{Idx(S_k)}|}$;
  7. $\mathscr{Y}_{r+1} = \frac{\sum_{k=1}^{K} |D_{Idx(S_k)}| \times \mathscr{Y}^k_{r+1}}{\sum_{k=1}^{K} |D_{Idx(S_k)}|}$;
end

Algorithm 2: Local Update Procedure of DFL

Input: i) E, # of local epochs;
ii) D, device dataset with hard labels Y;
iii) η, learning rate;
iv) R, # of total communication rounds;
v) $\mathcal{T}$, threshold of the loss function ratio;

DeviceUpdate($w, \mathscr{Y}$)**:**
1. Receive $(w, \mathscr{Y})$ from the cloud server;
2. $temp = w$;
for $e \leftarrow 1$ *to* E **do**
  3. $y \leftarrow \text{Prediction}(w, D)$;
  4. $\rho \leftarrow \text{Max}(1 - \frac{r}{R}, \mathcal{T})$;
  5. $\mathcal{L}(w) = \rho\ \mathcal{F}(y|w, Y) + (1 - \rho)\ \mathcal{G}(y|w, \mathscr{Y})$;
  6. $w = w - \eta \nabla \mathcal{L}(w)$;
end
7. $y \leftarrow \text{Prediction}(w, D)$;
8. $\mathscr{Y} \leftarrow \text{LabelWiseAverage}(y)$;
9. $\Delta = w - temp$;
10. Send $(\Delta, \mathscr{Y})$ to the cloud server;

where w and $\mathscr{Y}$ represent the model weight and the label-wise soft targets, respectively. K denotes the number of AIoT devices selected in each round of model training. Δ^k_{r+1} and $\mathscr{Y}^k_{r+1}$ indicate the model gradient and sample logits of device k in round $r+1$, $|D_k|$ represents the size of the data set D_k hosted by the kth AIoT device.

Algorithm 1 shows the key steps involved in our DFL algorithm. Step 1 initializes the global model and soft targets with w and $\mathscr{Y}$ using the function *Initialize*. In step 2, we calculate the number of selected AIoT devices K participating in each round of model training with the function *Max*, where C and N denote the fraction and the number of total AIoT devices. At the beginning of round r, step 3 randomly selects the devices participating in the model training of round r, where S is used to save the selected devices. Step 4 dispatches the global model and soft targets to all the selected devices in S. In step 5, all the selected devices upload both model gradients and the newly generated label-wise sample logits to the cloud server. Once the cloud server receives model gradients and label-wise sample logits from all the selected devices, steps 6 and 7 perform the aggregation.

Note that, since the aggregation process of soft targets is similar to the aggregation process of model gradients, any existing privacy protection approaches (e.g., secure aggregation [22] and variational Bayes-based method [23]) for FL can be easily integrated into our approach to avoid the risk of privacy exposure.

2) Local Update Procedure: When selected AIoT devices receive the latest global model w and soft targets $\mathscr{Y}$, they conduct the local update procedure. The local update procedure of our DFL approach involves two stages, i.e., the local training stage and the new label-wise sample logit generation stage. Similar to FedAvg, the predictions of our DFL model approximate the hard labels of local samples. To further improve the model accuracy, our approach makes the model predictions approximate to the soft targets related to the corresponding hard labels as well. Therefore, to make wisely use of the knowledge of both hard labels and soft targets, we design our loss function in model training as follows:

$$\mathcal{L}(w) = \rho\ \mathcal{F}(y|w, Y) + (1 - \rho)\ \mathcal{G}(y|w, \mathscr{Y}) \tag{7}$$

where $\mathcal{F}(y|w, Y)$ is the cross-entropy loss function, which denotes the distance between the prediction y and the corresponding hard label Y of the sample. Here, $\mathcal{G}(y|w, \mathscr{Y})$ is the Kullback–Leibler divergence loss function, which indicates the distance between the prediction y and the corresponding label-wise sample logits $\mathscr{Y}$ (extracted from the global soft targets) of the sample. The hyperparameter ρ ($\rho \in [0, 1]$) is the ratio of the two loss functions (see Section IV-C). Since the objective of local training is to minimize the loss function $\mathcal{L}(w)$, we can get the model update for each epoch as follows:

$$w = w - \eta \nabla \mathcal{L}(w) \tag{8}$$

where η denotes the learning rate and ∇ indicates the gradient. When the local training stage finishes, the new label-wise sample logit generation stage will be implemented. All the updated models perform predictions with local samples and calculate the label-wise sample logits. To improve communication efficiency, all the selected AIoT devices upload their model gradients (i.e., Δ) rather than the updated models to the cloud server for aggregation at the end of each round

$$\Delta^k_{r+1} = w^k_{r+1} - w^k_r. \tag{9}$$

Algorithm 2 presents the local update process of our DFL in detail. In steps 1 and 2 of the algorithm, AIoT devices receive the global model w and soft targets $\mathscr{Y}$ from the cloud server and save the received global model. Steps 3–6 show the implementation of the local training stage. At the beginning of each local epoch, step 3 makes predictions of data samples with the local model using the function *Prediction*. Step 4 calculates the current ratio of the two-loss functions ρ with the index r and the number R of training rounds and the threshold $\mathcal{T}$. Steps 5 and 6 iteratively update the model weight, where the

loss function is defined in (7). After the local training stage finishes, steps 7–9 generate the label-wise sample logits. To save network resources, step 9 calculates the model gradient for upload. Finally, step 10 uploads the model gradient and label-wise sample logits to the cloud server for aggregation.

C. Dynamic Adjustment Strategy

The ratio of the two-loss functions plays an important role in DFL local training since the weight of soft targets greatly impacts the model training. Generally, the knowledge of soft targets depends on the model accuracy, and the model accuracy increases as the number of training rounds increases. Therefore, there is insufficient knowledge of soft targets in the early stage of the model training process since the model is randomly initialized. In this case, the soft targets will make the model optimize in the wrong direction, which will slow down the model training. The knowledge of soft targets increases as the training continues, which can enhance the model with the knowledge that the hard labels do not have. However, the model training cannot rely mainly on soft targets according to [8]. Therefore, we need to set a threshold $\mathcal{T}$ to fix the ratio of the two-loss functions in the late stage of the model training process so that the model can achieve the best performance. To maximize the use of the soft targets and reduce their side effects, we design a dynamic adjustment strategy to control the loss function ratio as follows:

$$\rho = \mathrm{Max}\left(1 - \frac{r}{R}, \mathcal{T}\right) \tag{10}$$

where r and R represent the index of the current round and the total number of overall training rounds, respectively. $\mathcal{T}$ denotes the threshold to fix the ratio of the two-loss functions. As shown in (10), in the early stage of the model training process, the cross-entropy loss function is given a high proportion and gradually decreases with the number of training rounds, while the Kullback–Leibler divergence loss function is the opposite. The discussion about the optimal threshold is in the experimental part.

D. Convergence Analysis of Our DFL Approach

Similar to hard labels, the "global soft targets" can be used to represent features of all the data samples, which remain unchanged during the FL model training. As shown in (7), our approach combines both soft targets and hard labels as the approximation term of the FL model prediction. Due to the invariance of both soft targets $\mathscr{Y}$ and hard labels Y in (7), the convergence of DFL can be guaranteed. Inspired by the work in [24], we analyze the convergence rate of our DFL approach with two device participation scenarios (i.e., full device participation and partial device participation). We define the distributed optimization model of our DFL approach as follows:

$$\min_{w} \left\{ \Phi(w) \triangleq \sum_{k=1}^{N} p_k (\mathcal{F}_k(w) + \mathcal{G}_k(w)) \right\} \tag{11}$$

where N is the total number of all the AIoT devices, p_k is the probability of selecting the kth device such that $p_k \geq 0$ and $\sum_{k=1}^{N} p_k = 1$. $\mathcal{F}_k(w)$ and $\mathcal{G}_k(w)$ are two loss functions (i.e., the cross-entropy loss function and the Kullback–Leibler divergence loss function) which are defined as follows:

$$\begin{aligned} \mathcal{F}_k(w) &\triangleq \frac{1}{n_k} \sum_{j=1}^{n_k} \mathcal{F}\left(w; x_{k,j}\right) \\ \mathcal{G}_k(w) &\triangleq \frac{1}{n_k} \sum_{j=1}^{n_k} \mathcal{G}\left(w; x'_{k,j}\right) \end{aligned} \tag{12}$$

where n_k is the number of local samples in the kth device, $x_{k,j}$ is the local training samples concluding pictures and hard labels, $x'_{k,j}$ is the combination of the local samples and its corresponding soft targets.

Similar to [24], to analyze the convergence rate of our DFL approach, we make the following five assumptions on the functions $\mathcal{F}_1, \ldots, \mathcal{F}_N$ and $\mathcal{G}_1, \ldots, \mathcal{G}_N$.

Assumption 1: $\mathcal{F}_1, \ldots, \mathcal{F}_N$ and $\mathcal{G}_1, \ldots, \mathcal{G}_N$ are all $L-$*smooth*: for all v and w, $\mathcal{F}_k(v) \leq \mathcal{F}_k(w) + (v-w)^T \nabla \mathcal{F}_k(w) + (L/2)||v-w||^2$, $\mathcal{G}_k(v) \leq \mathcal{G}_k(w) + (v-w)^T \nabla \mathcal{G}_k(w) + (L/2)||v-w||^2$.

Assumption 2: $\mathcal{F}_1, \ldots, \mathcal{F}_N$ and $\mathcal{G}_1, \ldots, \mathcal{G}_N$ are all $\mu-$*strongly convex*: for all v and w, $\mathcal{F}_k(v) \geq \mathcal{F}_k(w) + (v-w)^T \nabla \mathcal{F}_k(w) + (\mu/2)||v-w||^2$, $\mathcal{G}_k(v) \geq \mathcal{G}_k(w) + (v-w)^T \nabla \mathcal{G}_k(w) + (\mu/2)||v-w||^2$.

Assumption 3: Let ξ_t^k and δ_t^k be sampled from the k^{th} device's local data uniformly at random. The variance of stochastic gradients in each device is bounded: $\mathbb{E}||\nabla \mathcal{F}_k(w_t^k, \xi_t^k) - \nabla \mathcal{F}_k(w_t^k)||^2 \leq \alpha_k^2$ for $k = 1, \ldots, N$ and $\mathbb{E}||\nabla \mathcal{G}_k(w_t^k, \delta_t^k) - \nabla \mathcal{G}_k(w_t^k)||^2 \leq \beta_k^2$ for $k = 1, \ldots, N$.

Assumption 4: The expected squared norm of stochastic gradients is uniformly bounded, i.e., $\mathbb{E}||\nabla \mathcal{F}_k(w_t^k, \xi_t^k)||^2 \leq G_1^2$ and $\mathbb{E}||\nabla \mathcal{G}_k(w_t^k, \delta_t^k)||^2 \leq G_2^2$ for all $k = 1, \ldots, N$ and $t = 1, \ldots, T-1$.

Assumption 5: From the tth local SGD, the distribution of soft targets no longer changes, so that $\mathcal{G}_k(w)$ is the only dependent variable of w.

Based on the assumptions above, we first analyze the convergence rate of our DFL approach with full device participation. The update of our DFL model can be described with the following formulas:

$$\begin{aligned} v_{t+1}^k &= w_t^k - \eta_t \left(\nabla \mathcal{F}_k\left(w_t^k, \xi_t^k\right) + \nabla \mathcal{G}_k\left(w_t^k, \delta_t^k\right) \right) \\ w_{t+1}^k &= \begin{cases} v_{t+1}^k, & \text{if} \quad T \nmid t+1 \\ \sum_{k=1}^{N} p_k v_{t+1}^k, & \text{if} \quad T | t+1 \end{cases} \end{aligned} \tag{13}$$

where w_t^k is the local model parameter maintained in the kth device at the tth SGD step, v_{t+1}^k is the immediate result of w_t^k with one step of SGD update. T is the local SGD steps within one training round. If $T|t+1$, our DFL activates all the AIoT devices. In our analysis, we define two virtual sequences

$$\overline{v}_t = \sum_{k=1}^{N} p_k v_t^k, \qquad \overline{w}_t = \sum_{k=1}^{N} p_k w_t^k. \tag{14}$$

By combining (13) and (14), we always have $\overline{v}_t = \overline{w}_t$. For convenience, we define $\overline{g}_t = \sum_{k=1}^{N} p_k [\nabla \mathcal{F}_k(w_t^k) + \nabla \mathcal{G}_k(w_t^k)]$ and $g_t = \sum_{k=1}^{N} p_k [\nabla \mathcal{F}_k(w_t^k, \xi_t^k) + \nabla \mathcal{G}_k(w_t^k, \delta_t^k)]$. Therefore, $\overline{v}_{t+1} = \overline{w}_t - \eta_t g_t$ and $\mathbb{E}[g_t] = \overline{g}_t$.

TABLE I
IID AND NON-IID DEVICE DATA SETTINGS FOR MNIST, CIFAR-10, AND CIFAR-100

Dataset	MNIST	CIFAR-10	CIFAR-100
Training Sample # in Total	60000	50000	50000
Training Sample # per AIoT Device	600	500	500
Label # in Total	10	10	20
Scenario	Data Setting		
IID	Uniform distribution		
non-IID	80% belong to one label, the remaining 20% belong to other labels		

Let $\Phi^\star$ be an optimal value of the loss function shown in (7), and $w^\star$ be its corresponding parameter. According to the lemmas presented in [24] and [25] (see proof details in the Appendix), we can get the inequality as follows:

$$\mathbb{E}\Big[\Phi(\overline{w}_t)\Big] - \Phi^\star \leq \frac{L}{2}\mathbb{E}\left\|\overline{w}_t - w^\star\right\|^2 \leq \frac{L}{2}\frac{v}{\gamma + t} \tag{15}$$

where

$$v = \max\left\{\frac{\beta^2 B}{2\beta\mu - 1}, (\gamma + 1)\Delta_1\right\} \tag{16}$$

and

$$B = 32(T-1)^2\left(G_1^2 + G_2^2\right)\sum_{k=1}^{N} p_k^2\left(\alpha_k^2 + \beta_k^2\right) + 8L\Gamma. \tag{17}$$

According to (15), our DFL converges to the global optimum at a rate of $O(1/t)$ for strongly convex and smooth functions. For the case of partial device participation, similar to [24], we can claim that the convergence rate of partial device participation is the same as that of full device participation.

V. EXPERIMENTAL RESULTS

A. Experimental Setup

To evaluate the effectiveness of our DFL approach, we implemented the approach on top of a cloud-based architecture consisting of a cloud server and a series of AIoT devices. Our DFL architecture was built on a workstation (with Intel i7-9700k CPU, 64-GB memory, NVIDIA GeForce GTX 2080Ti GPU), and ten Nvidia Jetson Nano boards (with ARM Cortex-A57 processor and 4-GB memory). Note that in the experiment, only 10 of the AIoT devices were emulated by the Jetson Nano boards, while the remaining devices were simulated on the workstation. The Jetson Nano boards connect to the workstation via a WiFi environment. Since not all devices are able to participate in each round of model training in the real AIoT application scenario, we set the fraction of AIoT devices to $C = 0.1$, i.e., ten devices were randomly selected to participate in each round of model training. For each AIoT device, we set the batch size, learning rate, and epoch of local training to 50, 0.01, and 5, respectively. For the performance comparisons of five methods, we set the threshold $\mathcal{T} = 0.6$ as an empirical optimal choice, which is detailed in Section V-C. Note that for the other hyperparameters of each baseline, we follow the parameters provided by this article authors.

We conducted experiments on four well-known benchmarks, i.e., MNIST, CIFAR-10, CIFAR-100 [26], and FEMNIST [27], respectively. In the experiments, we assumed that there are 100 AIoT devices for the first three benchmarks, respectively. Considering that all the AIoT devices are memory limited, we set the training samples of each benchmark equally to all the AIoT devices while putting the 10 000 test samples in the cloud server. In order to verify the model performance for different data distributions, we set two data scenarios (i.e., the IID scenario and the non-IID scenario) shown in Table I based on the Dirichlet Distribution according to [29]. For the IID scenario, all data samples were uniformly distributed on all the 100 AIoT devices. For the non-IID scenario, we set that 80% of the data samples on each device belong to one label, while the other 20% belong to other labels evenly. Note that the CIFAR-100 data set has two types of sample labels, i.e., the fine-grained label (100 classes) and the coarse-grained label (20 superclasses). According to the settings of our experimental scenario, we chose the coarse-grained labels as the sample categories to better distinguish the performance of different methods. For the data set FEMNIST from LEAF, we considered a non-IID scenario with 180 AIoT devices, where each device consists of more than 100 local samples.[1] Note that the raw data of FEMNIST is naturally non-IID distributed, involving class imbalance, data imbalance, and data heterogeneity.

To fairly validate the effectiveness of different methods, we conducted experiments using four randomly initialized models, i.e., CNN models used in [1], and three popular models (ResNet-20, VGG-16, and MobileNetV2) from Torchvision [28]. McMahan et al. [1] designed CNN models for MNIST and CIFAR-10. For the FEMNIST data set, we modified the output of the MNIST CNN model to 62, which is the labels of the samples. For the CIFAR-100 data set, we modified the output of the CIFAR-10 CNN model to 20, which is the coarse-grained labels of the samples. The Torchvision platform can provide the corresponding model interfaces according to the benchmarks we set. Therefore, the structure of these three models was fine-tuned according to different benchmarks.

The following sections first compare the performance of our proposed DFL with the state-of-the-art methods (i.e.,

[1]Using the command:./preprocess.sh -s niid –sf 0.05 -k 100 -t sample.

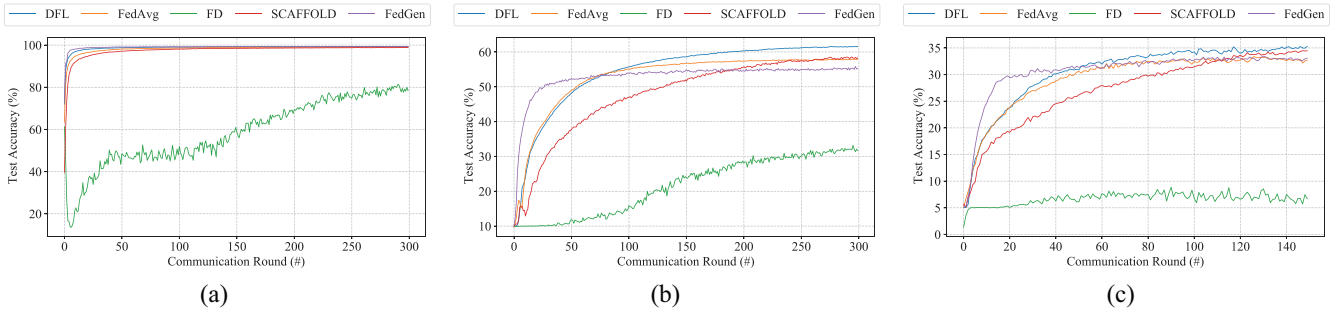

Fig. 3. Test accuracy comparison for the IID scenario using CNN. (a) MNIST. (b) CIFAR-10. (c) CIFAR-100.

TABLE II
TEST ACCURACY COMPARISON FOR THE IID SCENARIO USING FOUR MODELS

Dataset	Model	Test Accuracy of Different Methods(%)				
		FedAvg	FD	SCAFFOLD	FedGen	DFL (Ours)
MNIST	CNN	99.08	78.36	98.84	99.24	**99.33**
	ResNet-20	97.86	74.68	98.22	98.91	**98.93**
	VGG-16	99.13	88.40	98.79	99.19	**99.38**
	MobileNetV2	98.96	11.36	**99.35**	99.18	99.23
CIFAR-10	CNN	57.92	31.69	58.32	55.22	**61.48**
	ResNet-20	63.06	30.11	62.99	63.35	**64.18**
	VGG-16	79.81	33.39	81.63	80.27	**82.30**
	MobileNetV2	65.45	11.52	66.82	65.67	**69.64**
CIFAR-100	CNN	32.73	6.69	34.46	33.09	**35.28**
	ResNet-20	42.65	18.62	42.86	41.66	**43.16**
	VGG-16	55.21	5.06	55.39	55.73	**56.10**
	MobileNetV2	41.76	15.69	41.28	40.83	**42.85**

FedAvg [1], FD [11], SCAFFOLD [13], and FedGen [21]). Then, we investigate the impact of the loss function ratio and find the empirical optimal ratio of the two-loss functions with a series of experiments. To avoid the interference of random model initialization and out-of-order data set training on the experimental results, we ran each experiment ten times and took its mean value for a fair comparison.

B. Performance Evaluation

1) Performance Comparison for IID Scenarios: In the first experiment, we compared the performance of our method with four baseline methods using the IID scenario set in Table I. During the model training process of all the five methods, we tested the inference accuracy of the global models after each round of model aggregation in the cloud server. The model accuracy is equal to the ratio of the correctly predicted samples over the total testing samples using the cloud aggregated model. Due to the space limitation, we show the model accuracy trends using the CNN model on three benchmarks (i.e., MNIST, CIFAR-10, and CIFAR-100) along with the number of training rounds in Fig. 3. For each figure, the x-axis denotes the number of training rounds, and the y-axis indicates the model accuracy. Five curves with different colors represent the trends of the model inference accuracy of five different methods. From Fig. 3, we can find that the model accuracy of all the methods improves with the increase of training rounds. When the model accuracy does not increase significantly, we believe that the model converges. Since the FD method converges difficultly, we adaptively present the model convergence process of other methods in Fig. 3.

From Fig. 3, we can find that our DFL method achieves the highest model accuracy compared with the other four methods on all three benchmarks. Since our dynamic adjustment strategy gives the soft targets a small proportion in the early stage of model training, we can greatly reduce the side effects on model convergence caused by the insufficient knowledge of soft targets. We increase the proportion of the soft targets along with the training process, which can improve the model accuracy by maximizing the knowledge of soft targets. Therefore, our method can effectively improve the model inference accuracy without slowing down the model convergence rate. The model accuracy improvement of SCAFFOLD for the IID scenario is insignificant, and the model convergence speed of SCAFFOLD slows down. This is mainly because the added randomly initialized global variable misleads the optimization direction of the model in the early stage of model training. FedGen uses its built-in generators to generate extra samples, thereby speeding up model training. However, the samples generated by the generators of FedGen is naive, which will decrease the model accuracy in the late stage of the model training.

Table II presents the complete experimental results of the model accuracy of five methods. We tested the model accuracy of all the methods using four models on three benchmarks, and the highest model accuracy with the same model for the

TABLE III
TEST ACCURACY COMPARISON FOR THE NON-IID SCENARIO USING FOUR MODELS

Dataset	Model	Test Accuracy of Different Methods(%)				
		FedAvg	FD	SCAFFOLD	FedGen	DFL (Ours)
MNIST	CNN	98.70	23.27	98.42	99.08	**99.12**
	ResNet-20	95.91	35.31	96.07	96.90	**97.28**
	VGG-16	98.71	13.24	98.31	**99.33**	99.26
	MobileNetV2	98.34	11.05	98.45	98.44	**98.62**
CIFAR-10	CNN	51.48	11.53	52.81	48.86	**58.54**
	ResNet-20	50.86	20.55	**54.29**	50.64	53.03
	VGG-16	64.55	17.12	66.74	62.46	**72.18**
	MobileNetV2	38.01	10.66	39.66	30.94	**40.06**
CIFAR-100	CNN	27.54	5.59	30.53	28.65	**31.93**
	ResNet-20	17.62	7.31	23.48	23.92	**31.35**
	VGG-16	33.51	6.13	33.37	32.09	**35.21**
	MobileNetV2	17.41	6.13	19.26	22.14	**22.29**
FEMNIST	CNN	81.22	30.09	81.29	82.56	**84.83**
	ResNet-20	76.78	38.26	75.43	78.72	**81.93**
	VGG-16	83.50	26.38	83.07	82.49	**85.21**
	MobileNetV2	81.09	28.55	80.99	82.10	**83.17**

same data set is bolded. From Table II, we can find our DFL method achieves the highest model inference accuracy in 11 out of 12 cases. For example, when training the CNN model on the CIFAR-10 data set, the inference accuracy of FedAvg is 57.92%, while SCAFFOLD can achieve 58.32%, FedGen can achieve 55.22%, and our DFL can achieve 61.48%. This is mainly because the soft targets added by our method can improve the model inference accuracy effectively by enhancing the model knowledge. The global variables added in the SCAFFOLD method are based on the data distribution relationships among the AIoT devices to guide the model optimization direction of each AIoT device. Therefore, this method does not greatly improve the model inference accuracy for the IID scenario. Since the knowledge of the soft targets of FD is less than that of the model gradient, the FD model accuracy is lower than the FedAvg model accuracy. Note that the generators of FedGen can only generate simple data. Therefore, the model accuracy of this method becomes worse as the data set becomes more complex.

2) Performance Comparison for Non-IID Scenarios: To evaluate the performance of our DFL method for the non-IID scenario, we compared the five methods (i.e., FedAvg, FD, SCAFFOLD, and FedGen) using four benchmarks (i.e., MNIST, CIFAR-10, CIFAR-100, and FEMNIST), where the former three benchmarks follow the non-IID setting presented in Table I and the data set FEMNIST follows the non-IID setting provided by LEAF. Fig. 4 shows the trends of model accuracy using the CNN model along with the number of training rounds. Similar to the observations from Fig. 3, we can find that our approach outperforms the other four methods. Our DFL method achieves the highest model accuracy and the fastest model convergence speed on all four data sets.

Table III presents the complete experimental results of the model accuracy for the non-IID scenario. From Table III, we can find that our DFL method can achieve the best performance in 14 out of 16 cases. For example, when training

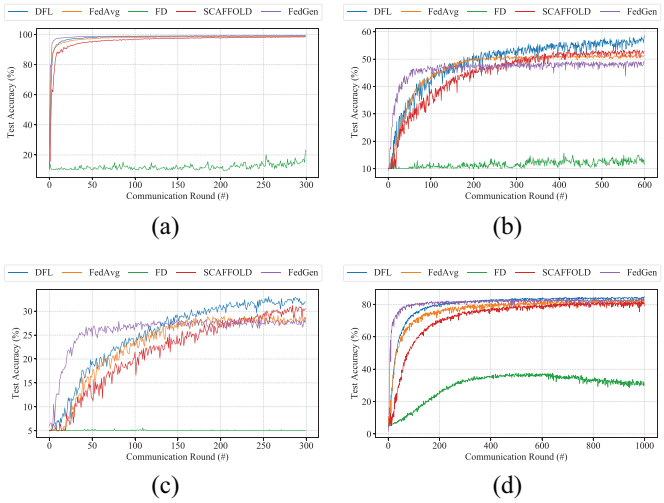

Fig. 4. Test accuracy comparison for the non-IID scenario using CNN. (a) MNIST. (b) CIFAR-10. (c) CIFAR-100. (d) FEMNIST.

the CNN model on the CIFAR-10 data set, our DFL method outperforms FedAvg, SCAFFOLD, and FedGen by 7.06%, 5.73%, and 9.68%, respectively. The reason why our approach is superior is mainly because the added soft targets can enhance model knowledge, which is effective for both IID and non-IID scenarios. Therefore, the local training process can use the knowledge of soft targets to improve the model accuracy. Note that the model accuracy of the FD method is 11.53% for this case, which is similar to that of a randomly initialized model. This is mainly because the knowledge of soft targets is insufficient to train a model. Therefore, the model trained by FD using soft targets alone is inaccurate.

3) Comparison of Communication Overhead: Table IV presents the size information for: 1) different neural network models for all the data sets and 2) the built-in generators

TABLE IV
SIZE OF MODELS AND GENERATORS

Model Name	Model Size (KB)				Generator Size (KB)
	MNIST	CIFAR-10	CIFAR-100	FEMNIST	
CNN	265.9	249.7	253.1	643.5	338.2
ResNet-20	908.7	909.9	1024	962.2	691.1
VGG-16	32768	137830.4	139366.4	32870.4	8806.4
MobileNetV2	9420.8	9420.8	9932.8	9728	2867.2

used by FedGen.[2] Note that the model size information in this table can be considered as the communication overhead of the classical FedAvg method. Due to additional global variables during the interaction between the cloud server and all the AIoT devices, the communication overhead of SCAFFOLD is always twice the overhead of FedAvg. During each round of FL training, FD dispatches and uploads only the soft targets, which is determined by the number of categories of data sets (i.e., 3.13 kB for CIFAR-100 with coarse-grained labels and 0.78 kB for the other data sets). Note that although FD requires much less communication overhead, its inference improvements on local models are limited due to the lack of model information, which strongly restricts its application in AIoT scenarios. Unlike the above four methods, the communication overhead of our DFL method equals the overhead sum of both FedAvg and FD, since the local training of DFL relies on both soft targets and model gradients. For example, when training the model ResNet-20 on data set FEMNIST using our DFL approach, the total size of both model gradients and soft targets is $(962.2 + 0.78) \times 2 = 1925.96$ kB, which requires 0.36 s on average for one DFL training round. However, SCAFFOLD needs 0.71 s for one training round, where the total size of both the global model and the global control variable is $(962.2 + 962.2) \times 2 = 3848.8$ kB. To conduct one round of FL training in this case, FedGen will lead to a communication overhead of $962.2+691.1+962.2 = 2615.5$ kB, which requires 0.48 s for the transmission of both model gradients and its built-in generator. Compared with the state-of-the-art methods (i.e., SCAFFOLD and FedGen), our DFL method results in much less communication overhead while the trained models can achieve higher accuracy.

C. Impacts of Dynamic Adjustment Strategy

Since the ratio (i.e., ρ) of the two-loss functions controls the proportion of hard labels and soft targets during the local training, it plays an important role in our DFL approach. To investigate the impacts of dynamic adjustment strategy, we conducted a series of experiments to verify the role of loss function ratio in different stages of model training. As a representative, Fig. 5 shows the trends of model accuracy of FedAvg and our DFL method with three different loss function ratio settings using the CNN model for the IID scenario of CIFAR-10. In Fig. 5, four curves with different colors represent the model accuracy trends of four methods, i.e., the DFL method with Fixed loss function Ratio named DFL-FR (marked in blue), the DFL method with Dynamic changing Ratio without the Threshold named DFL-DRw/T (marked in yellow), the DFL method with Dynamic changing Ratio and the Threshold named DFL-DRwT (marked in green), and FedAvg (marked in red).

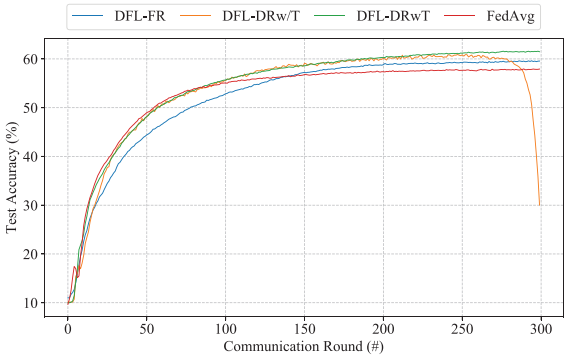

Fig. 5. Model accuracy trends of four methods.

From Fig. 5, we can find that the model accuracy of DFL-DRwT and DFL-DRw/T increases rapidly in the early stage of training while the DFL-FR model accuracy increases slowly. This is mainly because the knowledge of soft targets is insufficient in the early stage of model training, which can mislead the model optimization direction. The knowledge of soft targets increases as the model trains. In the late stage of training, there is sufficient knowledge of soft targets to guide the model training. Therefore, assigning a higher proportion to soft targets is beneficial to the model training as the number of training rounds increases. However, the model accuracy drops sharply in the final stage of the DFL-DRw/T model training. This is mainly because the proportion of hard labels is too small, and the model trained with soft targets alone is inaccurate. Therefore, we need to control the ratio between the two-loss functions to achieve the best model performance after the soft targets gain sufficient knowledge.

To investigate the empirical optimal threshold of the loss function ratio, we conducted experiments with thresholds from 0 to 1 with a step length of 0.1. Table V shows the experimental results of the model accuracy obtained on CIFAR-10 using different thresholds and our two data distribution settings, and the highest model accuracy is bolded. We can find that our DFL method achieves the highest model accuracy in 7 out of 8 cases when the threshold is set to 0.6. Only when MobileNetV2 is used for the IID scenario the model does not achieve the highest accuracy at $\mathcal{T} = 0.6$. In this case, the model obtains the highest accuracy at $\mathcal{T} = 0.5$, which is only 0.08 more than $\mathcal{T} = 0.6$. Therefore, to achieve the best model

[2]FedGen needs to dispatch both built-in generators and the global model, where the size of generators is shown in Table IV.

TABLE V
TEST ACCURACY COMPARISON WITH DIFFERENT THRESHOLDS

Scenario	Model	Test Accuracy (%)										
		T=0	T=0.1	T=0.2	T=0.3	T=0.4	T=0.5	T=0.6	T=0.7	T=0.8	T=0.9	T=1
IID	CNN	33.94	59.74	59.31	59.76	59.16	59.58	**61.48**	60.03	60.24	59.42	57.92
	ResNet-20	52.68	62.85	62.67	62.8	62.07	62.28	**64.18**	63.64	63.08	61.89	63.06
	VGG-16	79.13	80.98	81.55	81.67	81.63	81.18	**82.30**	82.23	81.24	80.12	79.81
	MobileNetV2	64.09	68.96	69.01	69.41	69.59	**69.72**	69.64	68.93	68.82	68.12	65.45
non-IID	CNN	18.45	51.35	51.57	51.27	52.37	53.34	**58.54**	54.26	55.29	54.38	51.48
	ResNet-20	24.65	27.74	36.03	39.76	45.67	46.55	**53.03**	52.44	50.34	51.59	50.86
	VGG-16	67.03	68.73	68.81	68.18	68.67	69.86	**72.18**	68.26	68.83	66.47	64.55
	MobileNetV2	33.89	35.18	34.01	37.48	38.85	39.96	**40.06**	39.82	35.88	38.34	38.01

performance, we set $\mathcal{T} = 0.6$ to maximize the use of soft targets and hard labels.

VI. CONCLUSION AND FUTURE WORK

Although FL techniques are becoming popular in AIoT applications, they are suffering from the problem of model inaccuracy. How to improve the model accuracy of FL under the limited network bandwidth and memory resources is becoming a major bottleneck in the design of AIoT applications. To address the above problem, this article presents a novel FL architecture based on KD named DFL, which can increase the model generalization ability. By adding soft targets to each round of model training, our proposed approach can increase the inference accuracy of the FL model without introducing significant communication and memory overhead. To further improve the performance of our DFL model, we designed a strategy to dynamically adjust the ratio of the two loss functions in KD to maximize the use of knowledge of soft targets. Comprehensive experimental results on four well-known benchmarks prove the effectiveness of our approach. For future work, we need to consider a better dynamic adjustment strategy, where the loss function ratio is controlled by the feedback of the model accuracy.

APPENDIX

A. Convergence Analysis of Our DFL Approach

Based on the assumptions we proposed in Section IV-D, we analyze the convergence of our DFL approach with full device participation. The update process of our DFL model can be formulated as follows:

$$v_{t+1}^k = w_t^k - \eta_t\left(\nabla\mathcal{F}_k\left(w_t^k, \xi_t^k\right) + \nabla\mathcal{G}_k\left(w_t^k, \delta_t^k\right)\right)$$
$$w_{t+1}^k = \begin{cases} v_{t+1}^k, & \text{if} \quad T \nmid t+1 \\ \sum_{k=1}^N p_k v_{t+1}^k, & \text{if} \quad T | t+1 \end{cases} \tag{18}$$

where w_t^k is the local model parameters maintained by the kth device at the tth SGD step, v_{t+1}^k is the immediate result of w_t^k with one step of SGD update. T is the local SGD steps within one training round. Here, we use the notation "$a|b$" to indicate that b can be divided exactly by a. If $T|t+1$, our DFL activates all the AIoT devices. In our analysis, we define two virtual sequences

$$\overline{v}_t = \sum_{k=1}^N p_k v_t^k, \qquad \overline{w}_t = \sum_{k=1}^N p_k w_t^k. \tag{19}$$

By combining (18) and (19), we always have $\overline{v}_t = \overline{w}_t$. For convenience, we define $\overline{g}_t = \sum_{k=1}^N p_k[\nabla\mathcal{F}_k(w_t^k) + \nabla\mathcal{G}_k(w_t^k)]$ and $g_t = \sum_{k=1}^N p_k[\nabla\mathcal{F}_k(w_t^k, \xi_t^k) + \nabla\mathcal{G}_k(w_t^k, \delta_t^k)]$. Therefore, $\overline{v}_{t+1} = \overline{w}_t - \eta_t g_t$ and $\mathbb{E}[g_t] = \overline{g}_t$.

Let $\Phi^\star$ be an optimal value of the loss function shown in (7), and $w^\star$ be its corresponding parameters, we have

$$\begin{aligned} \left\|\overline{v}_{t+1} - w^\star\right\|^2 &= \left\|\overline{w}_t - \eta_t g_t - w^\star - \eta_t\overline{g}_t + \eta_t\overline{g}_t\right\|^2 \\ &= \underbrace{\left\|\overline{w}_t - w^\star - \eta_t\overline{g}_t\right\|^2}_{A_1} + \underbrace{2\eta_t<\overline{w}_t - w^\star\eta_t\overline{g}_t, \overline{g}_t - g_t>}_{A_2} \\ &\quad + \underbrace{\eta_t^2\left\|g_t - \overline{g}_t\right\|^2}_{A_3}. \end{aligned} \tag{20}$$

Note that $\mathbb{E}A_2 = 0$. We next focus on bounding A_1. We divide A_1 into three terms

$$\begin{aligned} A_1 &= \left\|\overline{w}_t - w^\star - \eta_t\overline{g}_t\right\|^2 \\ &= \underbrace{\left\|\overline{w}_t - w^\star\right\|^2}_{} \underbrace{- 2\eta_t < \overline{w}_t - w^\star, \overline{g}_t >}_{B_1} + \underbrace{\eta_t^2||\overline{g}_t||^2}_{B_2}. \end{aligned} \tag{21}$$

We aim to bound B_1

$$\begin{aligned} B_1 &= -2\eta_t<\overline{w}_t - w^\star, \overline{g}_t> \\ &= -2\eta_t\sum_{k=1}^N p_k<\overline{w}_t - w^\star, \nabla\mathcal{F}_k\left(w_t^k\right) + \nabla\mathcal{G}_k\left(w_t^k\right)> \end{aligned} \tag{22}$$

where

$$\begin{aligned} &<\overline{w}_t - w^\star, \nabla\mathcal{F}_k\left(w_t^k\right) + \nabla\mathcal{G}_k\left(w_t^k\right)> \\ &= <\overline{w}_t - w_t^k, \nabla\mathcal{F}_k\left(w_t^k\right) > + < w_t^k - w^\star, \nabla\mathcal{F}_k\left(w_t^k\right)> \\ &\quad + <\overline{w}_t - w_t^k, \nabla\mathcal{G}_k\left(w_t^k\right) > + < w_t^k - w^\star, \nabla\mathcal{G}_k\left(w_t^k\right)>. \end{aligned} \tag{23}$$

By using the Cauchy–Schwarz inequality and the inequality of arithmetic and geometric means (AM-GMs), we can get

$$\begin{aligned} &-<\overline{w}_t - w_t^k, \nabla\mathcal{F}_k\left(w_t^k\right)> \\ &\le \frac{1}{2\eta_t}\left\|\overline{w}_t - w_t^k\right\|^2 + \frac{1}{2}\eta_t\left\|\nabla\mathcal{F}_k\left(w_t^k\right)\right\|^2 \\ &\quad - <w_t^k - w^\star, \nabla\mathcal{F}_k\left(w_t^k\right)> \\ &\le -\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k\left(w^\star\right)\right) - \frac{\mu}{2}\left\|w_t^k - w^\star\right\|^2 \\ &\quad - <\overline{w}_t - w_t^k, \nabla\mathcal{G}_k\left(w_t^k\right)> \end{aligned}$$

$$\leq \frac{1}{2\eta_t}\left\|\overline{w}_t - w_t^k\right\|^2 + \frac{1}{2}\eta_t\left\|\nabla\mathcal{G}_k\left(w_t^k\right)\right\|^2 - <w_t^k - w^\star, \nabla\mathcal{G}_k\left(w_t^k\right)> \leq -\left(\mathcal{G}_k\left(w_t^k\right) - \mathcal{G}_k\left(w^\star\right)\right) - \frac{\mu}{2}\left\|w_t^k - w^\star\right\|^2. \quad (24)$$

Therefore, B_1 can be presented as (25) based on (22), (23), and (24), i.e.,

$$\begin{aligned} B_1 \leq{}& \eta_t\sum_{k=1}^N p_k\left(\frac{1}{\eta_t}\left\|\overline{w}_t - w_t^k\right\|^2 + \eta_t\left\|\nabla\mathcal{F}_k\left(w_t^k\right)\right\|^2\right) \\ &- 2\eta_t\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k\left(w^\star\right) + \frac{\mu}{2}\left\|w_t^k - w^\star\right\|^2\right) \\ &+ \eta_t\sum_{k=1}^N p_k\left(\frac{1}{\eta_t}\left\|\overline{w}_t - w_t^k\right\|^2 + \eta_t\left\|\nabla\mathcal{G}_k\left(w_t^k\right)\right\|^2\right) \\ &- 2\eta_t\sum_{k=1}^N p_k\left(\mathcal{G}_k\left(w_t^k\right) - \mathcal{G}_k\left(w^\star\right) + \frac{\mu}{2}\left\|w_t^k - w^\star\right\|^2\right). \end{aligned} \quad (25)$$

By using Assumption 1, $\mathcal{F}_k(\cdot)$ and $\mathcal{G}_k(\cdot)$ can be bounded with the following formulas:

$$\begin{aligned} \left\|\nabla\mathcal{F}_k\left(w_t^k\right)\right\|^2 &\leq 2L\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k^\star\right) \\ \left\|\nabla\mathcal{G}_k\left(w_t^k\right)\right\|^2 &\leq 2L\left(\mathcal{G}_k\left(w_t^k\right) - \mathcal{G}_k^\star\right). \end{aligned} \quad (26)$$

Consequently, B_2 can be bounded using (27) based on the convexity of $||\cdot||^2$ and (26)

$$\begin{aligned} B_2 = \eta_t^2\left\|\overline{g}_t\right\|^2 &\leq \eta_t^2\sum_{k=1}^N p_k\left\|\nabla\mathcal{F}_k\left(w_t^k\right) + \nabla\mathcal{G}_k\left(w_t^k\right)\right\|^2 \\ &\leq 2\eta_t^2\sum_{k=1}^N p_k\left[\left\|\nabla\mathcal{F}_k\left(w_t^k\right)\right\|^2 + \left\|\nabla\mathcal{G}_k\left(w_t^k\right)\right\|^2\right] \\ &\leq 4L\eta_t^2\sum_{k=1}^N p_k\left[\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k^\star\right) + \left(\mathcal{G}_k\left(w_t^k\right) - \mathcal{G}_k^\star\right)\right]. \end{aligned} \quad (27)$$

Therefore, A_1 can be presented as (28) by combining (21), (25), and (27), i.e.,

$$\begin{aligned} A_1 \leq{}& (1 - 2\mu\eta_t)\left\|\overline{w}_t - w^\star\right\|^2 + 2\sum_{k=1}^N p_k\left\|\overline{w}_t - w_t^k\right\|^2 \\ &\underbrace{+6L\eta_t^2\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k^\star\right) + 6L\eta_t^2\sum_{k=1}^N p_k\left(\mathcal{G}_k\left(w_t^k\right) - \mathcal{G}_k^\star\right)}_{C} \\ &\underbrace{-2\eta_t\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k\left(w^\star\right)\right) - 2\eta_t\sum_{k=1}^N p_k\left(\mathcal{G}_k\left(w_t^k\right) - \mathcal{G}_k\left(w^\star\right)\right)}_{C}. \end{aligned} \quad (28)$$

Aiming to bound C, we define $\gamma_t = 2\eta_t(1 - 3L\eta_t)$, $\Gamma = \Phi^\star - \sum_{k=1}^N p_k\mathcal{F}_k^\star - \sum_{k=1}^N p_k\mathcal{G}_k^\star$. We split C into three terms

$$\begin{aligned} C ={}& -2\eta_t(1 - 3L\eta_t)\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k^\star\right) \\ &- 2\eta_t(1 - 3L\eta_t)\sum_{k=1}^N p_k\left(\mathcal{G}_k\left(w_t^k\right) - \mathcal{G}_k^\star\right) \\ &+ 2\eta_t\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w^\star\right) - \mathcal{F}_k^\star\right) + 2\eta_t\sum_{k=1}^N p_k\left(\mathcal{G}_k\left(w^\star\right) - \mathcal{G}_k^\star\right) \\ ={}& -\gamma_t\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \Phi^\star\right) - \gamma_t\sum_{k=1}^N p_k\left(\mathcal{G}_k\left(w_t^k\right) - \Phi^\star\right) \\ &+ (2\eta_t - \gamma_t)\sum_{k=1}^N p_k\left(\Phi^\star - \mathcal{F}_k^\star - \mathcal{G}_k^\star\right) - \gamma_t\Phi^\star \\ ={}& \underbrace{-\gamma_t\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \Phi^\star\right) - \gamma_t\sum_{k=1}^N p_k\left(\mathcal{G}_k\left(w_t^k\right) - \Phi^\star\right)}_{D} \\ &+ \underbrace{6L\eta_t^2\Gamma - \gamma_t\sum_{k=1}^N p_k\Phi^\star}. \end{aligned} \quad (29)$$

To bound D, we have

$$\begin{aligned} &\sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \Phi^\star\right) \\ &= \sum_{k=1}^N p_k\left(\mathcal{F}_k\left(w_t^k\right) - \mathcal{F}_k(\overline{w}_t)\right) + \sum_{k=1}^N p_k\left(\mathcal{F}_k(\overline{w}_t) - \Phi^\star\right) \\ &\geq \sum_{k=1}^N p_k < \bar{v}\mathcal{F}_k(\overline{w}_t), w_t^k - \overline{w}_t > + \sum_{k=1}^N p_k\mathcal{F}_k(\overline{w}_t) - \Phi^\star \\ &\geq -\frac{1}{2}\sum_{k=1}^N p_k\left[\eta_t\|\bar{v}\mathcal{F}_k(\overline{w}_t)\|^2 + \frac{1}{\eta_t}\left\|w_t^k - \overline{w}_t\right\|^2\right] + \sum_{k=1}^N p_k\mathcal{F}_k(\overline{w}_t) - \Phi^\star \\ &\geq -\sum_{k=1}^N p_k\left[\eta_t L\left(\mathcal{F}_k(\overline{w}_t) - \mathcal{F}_k^\star\right) + \frac{1}{2\eta_t}\left\|w_t^k - \overline{w}_t\right\|^2\right] + \sum_{k=1}^N p_k\mathcal{F}_k(\overline{w}_t) - \Phi^\star \end{aligned} \quad (30)$$

where the first inequality of (30) is derived based on the convexity of $\mathcal{F}_k$, the second inequality of (30) is from AM-GM inequality and the third inequality of (30) is derived based on (26). We use the same method to bound terms related to $\mathcal{G}_k$. Then, we have

$$\begin{aligned} D \leq{}& \gamma_t\sum_{k=1}^N p_k\left[\eta_t L\left(\mathcal{F}_k(\overline{w}_t) - \mathcal{F}_k^\star\right) + \frac{1}{2\eta_t}\left\|w_t^k - \overline{w}_t\right\|^2\right] \\ &- \gamma_t\left(\sum_{k=1}^N p_k\mathcal{F}_k(\overline{w}_t) - \Phi^\star\right) \\ &+ \gamma_t\sum_{k=1}^N p_k\left[\eta_t L\left(\mathcal{G}_k(\overline{w}_t) - \mathcal{G}_k^\star\right) + \frac{1}{2\eta_t}\left\|w_t^k - \overline{w}_t\right\|^2\right] \\ &- \gamma_t\left(\sum_{k=1}^N p_k\mathcal{G}_k(\overline{w}_t) - \Phi^\star\right). \end{aligned} \quad (31)$$

By combining (29) and (31), we can get

$$\begin{aligned}
C &\le \gamma_t(\eta_t L-1)\sum_{k=1}^{N}p_k\big(\mathcal{F}_k(\overline{w}_t)-\Phi^\star\big)\\
&\quad+\frac{\gamma_t}{2\eta_t}\sum_{k=1}^{N}p_k\left\|w_t^k-\overline{w}_t\right\|^2+\gamma_t\eta_t L\sum_{k=1}^{N}p_k\big(\Phi^\star-\mathcal{F}_k^\star\big)\\
&\quad+\gamma_t(\eta_t L-1)\sum_{k=1}^{N}p_k\big(\mathcal{G}_k(\overline{w}_t)-\Phi^\star\big)\\
&\quad+\frac{\gamma_t}{2\eta_t}\sum_{k=1}^{N}p_k\left\|w_t^k-\overline{w}_t\right\|^2+\gamma_t\eta_t L\sum_{k=1}^{N}p_k\big(\Phi^\star-\mathcal{G}_k^\star\big)\\
&\quad+6L\eta_t^2\Gamma-\gamma_t\sum_{k=1}^{N}p_k\Phi^\star\\
&\le \gamma_t(\eta_t L-1)\sum_{k=1}^{N}p_k\left(\mathcal{F}_k(\overline{w}_t)-\frac{1}{2}\Phi^\star\right)\\
&\quad+\gamma_t(\eta_t L-1)\sum_{k=1}^{N}p_k\left(\mathcal{G}_k(\overline{w}_t)-\frac{1}{2}\Phi^\star\right)\\
&\quad+\frac{\gamma_t}{\eta_t}\sum_{k=1}^{N}p_k\left\|w_t^k-\overline{w}_t\right\|^2+\gamma_t\eta_t L\sum_{k=1}^{N}p_k\left(\frac{1}{2}\Phi^\star-\mathcal{F}_k^\star\right)\\
&\quad+\gamma_t\eta_t L\sum_{k=1}^{N}p_k\left(\frac{1}{2}\Phi^\star-\mathcal{G}_k^\star\right)+6L\eta_t^2\Gamma\\
&\quad-\gamma_t(\eta_t L-1)\Phi^\star+\gamma_t\eta_t L\Phi^\star-\gamma_t\Phi^\star\\
&= \gamma_t(\eta_t L-1)\sum_{k=1}^{N}p_k\big(\mathcal{F}_k(\overline{w}_t)+\mathcal{G}_k(\overline{w}_t)-\Phi^\star\big)\\
&\quad+\frac{\gamma_t}{\eta_t}\sum_{k=1}^{N}p_k\left\|w_t^k-\overline{w}_t\right\|^2+\gamma_t\eta_t L\big(\Phi^\star-\mathcal{F}_k^\star-\mathcal{G}_k^\star\big)+6L\eta_t^2\Gamma\\
&\le 2\sum_{k=1}^{N}p_k\left\|w_t^k-\overline{w}_t\right\|^2+\Big(6L\eta_t^2+\gamma_t\eta_t L\Big)\Gamma\\
&\le 2\sum_{k=1}^{N}p_k\left\|w_t^k-\overline{w}_t\right\|^2+8L\eta_t^2\Gamma. \qquad (32)
\end{aligned}$$

In the last inequality of (32), we use the following three facts: 1) $\eta_t L-1\le -(3/4)\le 0$ and $\sum_{k=1}^{N}p_k(\mathcal{F}_k(\overline{w}_t)+\mathcal{G}_k(\overline{w}_t)-\Phi^\star)=\Phi(\overline{w}_t)-\Phi^\star\ge 0$; 2) $\Gamma\ge 0$ and $6L\eta_t^2+\gamma_t\eta_t L\le 8\eta_t^2 L$; and 3) $(\gamma_t/2\eta_t)\le 1$. Recalling the expression of A_1 and plugging C into it, we have

$$\begin{aligned}
A_1 &\le (1-2\mu\eta_t)\left\|\overline{w}_t-w^\star\right\|^2\\
&\quad+4\sum_{k=1}^{N}p_k\left\|\overline{w}_t-w_t^k\right\|^2+8L\eta_t^2\Gamma. \qquad (33)
\end{aligned}$$

According to Assumption 3, the variance of the stochastic gradients $\mathcal{F}$ and $\mathcal{G}$ in device k is bounded by α_k^2 and β_k^2. Consequently, we have

$$\begin{aligned}
&\mathbb{E}\left\|g_t-\overline{g}_t\right\|^2\\
&=\mathbb{E}\left\|\sum_{k=1}^{N}p_k\Big(\nabla\mathcal{F}_k\Big(w_t^k,\xi_t^k\Big)+\nabla\mathcal{G}_k\Big(w_t^k,\xi_t^k\Big)\right.\\
&\qquad\left.-\nabla\mathcal{F}_k\Big(w_t^k\Big)-\nabla\mathcal{G}_k\Big(w_t^k\Big)\Big)\right\|^2\\
&=\mathbb{E}\left\|\sum_{k=1}^{N}p_k\Big(\nabla\mathcal{F}_k\Big(w_t^k,\xi_t^k\Big)-\nabla\mathcal{F}_k\Big(w_t^k\Big)\Big)\right.\\
&\qquad\left.+\sum_{k=1}^{N}p_k\Big(\nabla\mathcal{G}_k\Big(w_t^k,\xi_t^k\Big)-\nabla\mathcal{G}_k\Big(w_t^k\Big)\Big)\right\|^2\\
&=\sum_{k=1}^{N}p_k^2\mathbb{E}\left\|\nabla\mathcal{F}_k\Big(w_t^k,\xi_t^k\Big)-\nabla\mathcal{F}_k\Big(w_t^k\Big)\Big)\right\|^2\\
&\qquad+\sum_{k=1}^{N}p_k^2\mathbb{E}\left\|\nabla\mathcal{G}_k\Big(w_t^k,\delta_t^k\Big)-\nabla\mathcal{G}_k\Big(w_t^k\Big)\Big)\right\|^2\\
&\le\sum_{k=1}^{N}p_k^2\Big(\alpha_k^2+\beta_k^2\Big). \qquad (34)
\end{aligned}$$

Since our DFL requires a communication round each T SGD steps, for any $t\ge 0$, there exists a $t_0\le t$, such that $t-t_0\le T-1$ and $w_{t_0}^k=\overline{w}_{t_0}$ for all $k=1,2,\ldots,N$. Based on the fact that η_t is nonincreasing and $\eta_{t_0}\le 2\eta_t$ for all $t-t_0\le T-1$, we can get

$$\begin{aligned}
&\mathbb{E}\sum_{k=1}^{N}p_k\left\|\overline{w}_t-w_t^k\right\|^2\\
&=\mathbb{E}\sum_{k=1}^{N}p_k\left\|\Big(w_t^k-\overline{w}_{t_0}\Big)-\Big(\overline{w}_t-\overline{w}_{t_0}\Big)\right\|^2\\
&\le\mathbb{E}\sum_{k=1}^{N}p_k\left\|w_t^k-\overline{w}_{t_0}\right\|^2\\
&\le\sum_{k=1}^{N}p_k\mathbb{E}\sum_{t=t_0}^{t-1}(T-1)\eta_t^2\left\|\nabla\mathcal{F}_k\Big(w_t^k,\xi_t^k\Big)+\nabla\mathcal{G}_k\Big(w_t^k,\delta_t^k\Big)\right\|^2\\
&\le\sum_{k=1}^{N}p_k\mathbb{E}\sum_{t=t_0}^{t-1}(T-1)\eta_t^2 2\left[\left\|\nabla\mathcal{F}_k\Big(w_t^k,\xi_t^k\Big)\right\|^2+\left\|\nabla\mathcal{G}_k\Big(w_t^k,\delta_t^k\Big)\right\|^2\right]\\
&\le 2\sum_{k=1}^{N}p_k\sum_{t=t_0}^{t-1}(T-1)\eta_{t_0}^2\Big(G_1^2+G_2^2\Big)\\
&\le 2\sum_{k=1}^{N}p_k\eta_{t_0}^2(T-1)^2\Big(G_1^2+G_2^2\Big)\\
&\le 8\eta_t^2(T-1)^2\Big(G_1^2+G_2^2\Big). \qquad (35)
\end{aligned}$$

Therefore, we can obtain (36) by combining (33)–(35)

$$\begin{aligned}
&\mathbb{E}||\overline{w}_{t+1}-w^\star||^2\le(1-2\mu\eta_t)\mathbb{E}||\overline{w}_t-w^\star||^2+\eta_t^2 B\\
&B=32(T-1)^2\Big(G_1^2+G_2^2\Big)\sum_{k=1}^{N}p_k^2\Big(\alpha_k^2+\beta_k^2\Big)+8L\Gamma. \qquad (36)
\end{aligned}$$

For a diminishing stepsize similar to [24], $\eta_t=(\beta/t+\gamma)$ for some $\beta>(1/\mu)$ and $\gamma>0$ such that $\eta_1\le\min\{(1/\mu),(1/4L)\}=(1/4L)$ and $\eta_t\le 2\eta_{t+T}$. Let $\Delta_t=\mathbb{E}||\overline{w}_t-w^\star||^2$. Based on (36), we have

$$\Delta_{t+1}\le(1-2\eta_t\mu)\Delta_t+\eta_t^2 B. \qquad (37)$$

According to [24], $\Delta_t\le(v/\gamma+t)$, where $v=\max\{([\beta^2 B]/[2\beta\mu-1]),(\gamma+1)\ \Delta_1\}$.

Then, by the L-smoothness of $\Phi(\cdot)$, we can get

$$\mathbb{E}\Big[\Phi(\overline{w}_t)\Big]-\Phi^\star\le\frac{L}{2}\mathbb{E}\left\|\overline{w}_t-w^\star\right\|^2\le\frac{L}{2}\frac{v}{\gamma+t} \qquad (38)$$

where

$$v = \max\left\{\frac{\beta^2 B}{2\beta\mu - 1}, (\gamma + 1)\Delta_1\right\} \tag{39}$$

and

$$B = 32(T-1)^2\left(G_1^2 + G_2^2\right)\sum_{k=1}^{N} p_k^2\left(\alpha_k^2 + \beta_k^2\right) + 8L\Gamma. \tag{40}$$

According to (38), our DFL converges to the global optimum at a rate of $O(1/t)$ for strongly convex and smooth functions given L, v, and γ are constants. For the case of partial device participation, similar to [24], we can claim that the convergence rate of partial device participation is the same as that of full device participation.

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 *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[2] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. Int. Conf. Comput. Stat. (COMPSTAT)*, 2010, pp. 177–186.

[3] K. Bonawitz et al., "Towards federated learning at scale: System design," in *Proc. Int. Conf. Mach. Learn. Syst. (MLSys)*, 2019, pp. 1–15.

[4] H. Yang et al., "Lead federated neuromorphic learning for wireless edge artificial intelligence," *Nat. Commun.*, vol. 13, p. 4269, Jul. 2022.

[5] J. Xia, T. Liu, Z. Ling, T. Wang, X. Fu, and M. Chen, "PervasiveFL: Pervasive federated learning for heterogeneous IoT systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4100–4111, Nov. 2022.

[6] P. Li, Z. Chen, L. Yang, Q. Zhang, and M. J. Deen, "Deep convolutional computation model for feature learning on big data in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 790–798, Feb. 2018.

[7] W. Y. B. Lim et al., "When information freshness meets service latency in federated learning: A task-aware incentive scheme for smart industries," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 457–466, Jan. 2022.

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

[9] R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. Dahl, and G. Hinton, "Large scale distributed neural network training through online distillation," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–12.

[10] F. Sattler, A. Marbán, R. Rischke, and W. Samek, "Communication-efficient federated distillation," 2020, *arXiv:2012.00632*.

[11] 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*.

[12] D. Sui, Y. Chen, J. Zhao, Y. Jia, Y. Xie, and W. Sun, "FedED: Federated learning via ensemble distillation for medical relation extraction," in *Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, 2020, pp. 2118–2128.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," 2019, *arXiv:1910.06378*.

[14] Y. Huang et al., "Personalized cross-silo federated learning on non-IID data," in *Proc. Appl. Artif. Intell. (AAAI)*, 2021, pp. 7865–7873.

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

[16] M. Xie et al., "Multi-center federated learning," 2020, *arXiv:2005.01026*.

[17] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," 2020, *arXiv:2004.11791*.

[18] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1–13.

[19] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[20] 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," 2020, *arXiv:2008.06180*.

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

[22] K. A. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2017, pp. 1175–1191.

[23] M. Park, J. R. Foulds, K. Chaudhuri, and M. Welling, "Variational bayes in private settings," *J. Artif. Intell. Res.*, vol. 68, pp. 109–157 May 2020.

[24] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on Non-IID data," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–12.

[25] H. Kim and C. Tepedelenlioglu, "Performance bounds on average error rates using the AM-GM inequality and their applications in relay networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 8, pp. 2986–2995, Aug. 2012.

[26] "TorchvisionData, dataset of MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100." 2019. [Online]. Available: https://pytorch.org/docs/stable/torchvision/datasets.html

[27] S. Caldas et al., "LEAF: A benchmark for federated settings," 2018, *arXiv:1812.01097*.

[28] "TorchvisionModel, general models from torchvision." 2019. [Online]. Available: https://pytorch.org/docs/stable/torchvision/models.html

[29] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019, *arXiv:1909.06335*.

Tian Liu received the B.S. and M.E. degrees from the Department of Computer Science and Technology, Hohai University, Nanjing, China, in 2011 and 2014, respectively, the Engineer degree from the Department of Information and Statistic, Polytech-Lille, Villeneuve-d'Ascq, France, in 2012, and the Ph.D. degree from the Software Engineering Institute, East China Normal University, Shanghai, China, in 2022.

He is a Lecturer with the Department of Information Science and Engineering, Zaozhuang University, Zaozhuang, China. His research interests are in the area of federated learning, machine learning, Internet of Things, and cloud computing.

Jun Xia (Graduate Student Member, IEEE) received the B.S. degree from the Department of Computer Science and Technology, Hainan University, Hainan, China, in 2016, and the M.E. degree from the Department of Computer Science and Technology, Jiangnan University, Wuxi, China, in 2019, respectively. He is currently pursuing the Ph.D. degree with the Software Engineering Institute, East China Normal University, Shanghai, China.

His research interests are in the area of federated learning, AIoT applications, cloud computing, and heterogeneous computing.

Zhiwei Ling received the B.S. degree from the Department of Education Information Technology, East China Normal University, Shanghai, China, in 2021, where he is currently pursuing the master's degree with the Software Engineering Institute.

His research interests are in the area of federated learning, machine learning, and Internet of Things.

Xin Fu (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Florida, Gainesville, FL, USA, in 2009.

She was an NSF Computing Innovation Fellow with the Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL, USA, from 2009 to 2010. From 2010 to 2014, she was an Assistant Professor with the Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA. She is currently an Associate Professor with the Electrical and Computer Engineering Department, University of Houston, Houston, TX, USA. Her research interests include high-performance computing, machine learning, energy-efficient computing, and mobile computing.

Dr. Fu is a recipient of the 2014 NSF Faculty Early CAREER Award, the 2012 Kansas NSF EPSCoR First Award, and the 2009 NSF Computing Innovation Fellow.

Shui Yu (Senior Member, IEEE) received the Ph.D. degree from Deakin University, Melbourne, VC, Australia, in 2004.

He currently is a Professor with the School of Computer Science, University of Technology Sydney, Sydney, NSW, Australia. He has published four monographs and edited two books, more than 400 technical papers, including top journals and top conferences, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, and INFOCOM. His h-index is 63. He initiated the research field of networking for big data in 2013, and his research outputs have been widely adopted by industrial systems, such as Amazon cloud security. His research interest includes big data, security and privacy, networking, and mathematical modeling.

Dr. Yu is a Distinguished Visitor of IEEE Computer Society, a Voting Member of IEEE ComSoc Educational Services board, and an Elected Member of Board of Governor of IEEE Vehicular Technology Society. He is currently serving a number of prestigious editorial boards, including IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (Area Editor), *IEEE Communications Magazine*, and IEEE INTERNET OF THINGS JOURNAL. He served as a Distinguished Lecturer of IEEE Communications Society from 2018 to 2021.

Mingsong Chen (Senior Member, IEEE) received the B.S. and M.E. degrees from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer engineering from the University of Florida, Gainesville, FL, USA, in 2010.

He is currently a Professor with the Software Engineering Institute, East China Normal University, Shanghai, China. His research interests are in the area of cloud computing, design automation of cyber–physical systems, parallel and distributed systems, and formal verification techniques.

Dr. Chen is an Associate Editor of *IET Computers & Digital Techniques* and *Journal of Circuits, Systems, and Computers*.