FedDKD: Federated learning with decentralized knowledge distillation

Xinjia Li · Boyu Chen · Wenlian Lu

Accepted: 25 December 2022 / Published online: 3 February 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
The heterogeneity of the data distribution generally influences federated learning performance in neural networks. For a well-performing global model, taking a weighted average of the local models, as in most existing federated learning algorithms, may not guarantee consistency with local models in the space of neural network maps. In this paper, we highlight the significance of the space of neural network maps to relieve the performance decay produced by data heterogeneity and propose a novel federated learning framework equipped with the decentralized knowledge distillation process (FedDKD). In FedDKD, we introduce a decentralized knowledge distillation (DKD) module to distill the knowledge of local models to teach the global model approaching the neural network map average by optimizing the divergence defined in the loss function, other than only averaging parameters as in the literature. Numerical experiments on various heterogeneous datasets reveal that FedDKD outperforms the state-of-the-art methods, especially on some extremely heterogeneous datasets.

Keywords Federated learning · Knowledge distillation · Heterogeneous data · Data-free algorithm

1 Introduction

Data privacy and security have attracted increasing attention with the widespread application of deep learning in real life, such as smartphones, Internet of Things (IoT) devices, and digital health. Thus, federated learning (FL) [1–5], which focuses on the challenge of training a global model while leaving the users’ private data on their device, has been widely used in rich development applications such as smart grids [6], medical diagnostic image analysis [7], and IoT perception data prediction [8].

The typical federated learning methods, including FedAvg [1], FedProx [9], and FedMAX [10], follow the framework proposed by FedAvg. The server iteratively collects parameters or gradients from local models and updates the global model weights with the weighted average of the local models. Generally, federated learning employs stochastic gradient descent (SGD) to optimize the local models on the clients. It is well known that the SGD requests independently identically distributed samples to guarantee that the stochastic gradient is an unbiased estimate of the full gradient [11]. However, the datasets are heterogeneously distributed on clients in many situations where they may have different data sources (e.g., medical radiology images from various hospitals [12]) or come from the same source but have heterogeneous label distributions [1, 9, 13–15].

Data heterogeneity is one of the primary challenges in federated learning [5]. Previous literature has shown that federated learning performance is significantly reduced by data heterogeneity [14], which also harms the convergence speed of the algorithms [16].

Another primary challenge is neural network overparameterization. It has been argued that averaging the parameters...
is not optimal for obtaining the global model. For instance, in FedMA [13], the authors argued that the permutation invariance of neural network architectures partially explains why averaging the local parameters is a naive and not optimal method for obtaining global parameters, despite its practical performance [1, 10]. Consider a basic fully connected neural network \( \hat{y} = \sigma(xW^{(1)}\Pi)\Pi W^{(2)} \), where \( \{W^{(1)}, W^{(2)}\} \) are optimal weights, and \( \Pi \) is a permutation matrix [13]. If both local models respectively obtain the optimal weights \( \{W^{(1)}\Pi_j, \Pi_j W^{(2)}\} \) and \( \{W^{(1)}\Pi_j', \Pi_j' W^{(2)}\} \), then averaging the weights makes it almost impossible to obtain optimal weights given that \( \Pi_j \neq \Pi_j' \). However, the optimal global model is still achievable if the two local models are averaged in the function space (i.e., averaging the same two function mappings in this example).

Motivated by these findings, we experiment on CIFAR-10 and CIFAR-100 to determine whether the averaging model in the parameter space or the function space is more favorable for building a global model in a data heterogeneity situation. Following the previous work [13], we divide CIFAR-10 into \( K = 16 \) clients by sampling \( p_r \sim \text{Dir}_K(\alpha) \) from a Dirichlet distribution and allocating a \( p_{r,k} \) proportion of the training instances of class \( r \) to the local client \( k \). The distributions of CIFAR-10 for \( \alpha = 0.05, 0.1, 0.5 \) are shown in Fig. 1. The distributions of CIFAR-100 are provided in Supplementary Materials A.

Obtaining the well-trained models on clients, we average the local models in the parameter space as done in FedAvg and also average the models in the function space by collecting the weighted average of the local test accuracy in local test sets and then comparing their performance. As shown in Fig. 2, when local data are more heterogeneously distributed on clients, the ideal averaged global model in the function space is better than the averaged model in the parameter space, which is consistent with our common intuition. This motivates us to go further based on the parameter-averaged model to approach the ideal averaged model in the function space and shorten the gap between the parameter-averaged model and the function-averaged model in data heterogeneity situations.

In this paper, we propose a novel federated learning scheme (Fig. 3), FedDKD, which introduces a module of decentralized knowledge distillation (DKD) to average the local models in the function space instead of the parameter space. Without shared data on the server, the DKD enables the global model to integrate knowledge learned from local datasets to approach the average of local models in the function space. Unlike the existing federated learning methods with knowledge distillation, FedDKD is a data-
The test performance gap between the parameter-averaged and function-averaged models with different data heterogeneity on CIFAR-10 (left) and CIFAR-100 (right). Keeping the same communication round, when \( \alpha \) is smaller (i.e., local datasets are more heterogeneously distributed on clients), the test performance gain of the function-averaged model compared to the parameter-averaged model is larger.

FedDKD: Federated learning with decentralized knowledge distillation

Different from other data-free federated learning methods based on distillation learning which distill the global model knowledge for the local models separately (Fig. 4a),...
FedDKD jointly distills the local model knowledge for the global model and updates the global model on the server, as shown in Fig. 4b, to obtain the average in function space rather than that in parameter space. Our FedDKD can efficiently eliminate the damage caused by the heterogeneity of local datasets and help determine a better global model in the function space than directly taking the weighted average of local models.

Numerical experiments on diverse heterogeneous datasets demonstrate the effectiveness of this method for achieving better test accuracy. Moreover, FedDKD achieves training and communication efficiency on some extremely heterogeneous datasets. The DKD module can work as an additional plug-in module with diverse state-of-the-art federated learning technologies to improve performance.

Specifically, our contributions consist of the following:

1. Unlike the existing methods focusing on averaging in parameter space, we highlight the significance of the neural network map space, which is more suitable for scenarios with heterogeneous datasets.
2. We propose a novel federated learning scheme with a decentralized knowledge distillation module. This is the first algorithm to distill knowledge from local models to the global model using the view of function map averaging.
3. The proposed DKD module demonstrates convenience as a plug-in into other parameter-averaged methods to achieve better performance for heterogeneous datasets.
4. Compared with various methods, including FedAvg [1], FedProx [9], FedMAX [10], MOON [17], FedDistill [18], FedGEN [19] and FedBN [12], FedDKD can achieve better performance with the same communication cost and fewer local training steps for various datasets with various types of heterogeneity.

The paper is organized as follows. Section 2 reviews the most related works and presents the difference between FedDKD and other methods. Section 3 elaborates on the details of our method. Section 4 presents experiments on diverse datasets to validate the effectiveness of our method. In Section 5, we discuss the relation to the parameter-averaged model and how to extend FedDKD to second-order methods, and we summarize our works in Section 6.

## 2 Related works

This section briefly describes the most related works of knowledge distillation and federated learning with heterogeneous data. In addition, the difference between our method and the previous studies based on knowledge distillation is introduced.

### 2.1 Knowledge distillation

Knowledge distillation (KD) defines a learning schema to transfer knowledge from a larger teacher network to a smaller student network [20]. By optimizing the student model to match logits or intermediate activations of the teacher model, KD can compress the knowledge from a powerful teacher model to improve model efficiency.
FedDKD: Federated learning with decentralized knowledge distillation

with multiple teachers by multi-task learning. can be considered an extension of knowledge distillation
In this paper, we employ KD in federated learning, which
an essential prerequisite for the global model. In addition,
public dataset and attempt to mitigate the effects of data
global parameters. These data-free methods do not need any
other clients’ information and followed FedAvg to obtain
bias on local models by considering the loss involving
FedDistill [18] and FedGen [19], have introduced inductive
with clients in applications.

3.1 Decentralized knowledge distillation module
In this section, we propose an approach based on knowledge
distillation to approach the averaged model in the function
space.

As [36] does, we use a generalized definition in this
paper, which requires loss function is also in a divergence
form. More specifically, we need the loss function
\int(x) = div(\cdot, \cdot) : \mathbb{R}^C \times \mathbb{R}^C \to \mathbb{R}^+ to be a divergence, i.e., a
differentiable binary function satisfied that

\begin{align*}
div(u, v) &\geq 0, \forall u, v \\
div(u, u) &= 0, \forall u.
\end{align*}
Assume that a global ground truth map $\Phi^*(x)$ exists, providing the ground truth label $y$ (i.e., $y = \Phi^*(x)$). We attempt to build a parametric global model $\Phi(x; w)$ with parameters $w$ to approximate the ideal $\Phi^*(x)$ by optimizing the following objective function:

$$
\min_w \int div(\Phi^*(x), \Phi(x; w)) P(dx),
$$

(1)

where $div(\cdot, \cdot)$ refers to some divergence to measure the distance between $\Phi(x; w)$ and the ideal $\Phi^*(x)$, such as the total variation distance $\delta(\cdot, \cdot)$. In the federated learning framework, local model $\Phi(x; w_k), k \in \{1, \cdots, K\}$, which has the same architecture as the global model $\Phi(x; w)$, is trained on local dataset $D_k$. We set the integrated model on clients as follows:

$$
T(x) := \sum_{k=1}^{K} \Phi(x; w_k) \mathbb{1}_{D_k}(x),
$$

(2)

where $\mathbb{1}$ is the indicator function.

When the triangular inequality holds for some $div(\cdot, \cdot)$ (i.e., $div(u, v) \leq div(u, h) + div(h, v)$ for all $u, v,$ and $h$ it is defined with the same neural network permitted), we can provide the upper bound of the objective function (1) as follows:

$$
\int div(\Phi^*(x), \Phi(x; w)) P(dx)
\leq \int (div(\Phi^*(x), T(x)) + div(T(x), \Phi(x; w))) P(dx)
\leq \sum_{k=1}^{K} q_k \int_{D_k} div(\Phi^*(x), \Phi(x; w_k)) P_k(dx)
\leq L_1 + L_2,
$$

(3)

where

$$
L_1 = \sum_{k=1}^{K} q_k \int_{D_k} D_{KL}(\Phi^*(x)\|\Phi(x; w_k)) P_k(dx),
$$

$$
L_2 = \sum_{k=1}^{K} q_k \int_{D_k} D_{KL}(\Phi(x; w_k)\|\Phi(x; w)) P_k(dx)
$$

(5)

and $D_{KL}(\cdot \| \cdot)$ is the KL divergence. The proof is provided in Supplementary Materials C.

Motivation The following inequality above indicates that minimizing the triangular upper bounds is reasonable instead of minimizing (1) in federated learning. Specifically, minimizing $L_1$ in (3) is equivalent to the following

$$
\min_w \int_{D_k} div(\Phi(x; w_k), \Phi^*(x)) P_k(dx),
$$

(6)

that trains the local model on client $k$ with the local dataset $D_k$. Minimizing $L_2$ in (3) is the decentralized knowledge distillation where the global model $\Phi(x; w)$ learns the knowledge of local trained models $\Phi(x; w_k), k = 1, \cdots, K$:

$$
\hat{w} = \arg \min_w L_2
$$

(6)

that provides the center of all local models $\Phi(\cdot; w_k)$ in the function space. Here we call this distillation process the DKD module.

3.2 The proposed approach: FedDKD

This section discusses how the DKD module works in federated learning, as shown in Fig. 3. Consider a typical federated learning problem with the disjoint datasets $D_k := \{(x_{k,i}, y_{k,i})\}_{i=1}^{n_k}$, where $(x_{k,i}, y_{k,i}) \in \mathbb{R}^d \times \mathbb{R}$ on client $k$. The local model $\Phi(x; w_k)$ on client $k$ and the global model $\Phi(x; w)$ are defined with the same neural network architecture.

As mentioned in Section 3.1, Proposition 1 notes that with the square of total variation distance measurement, (1) can be optimized by minimizing the upper bound $L_1' + L_2'$ based on the KL divergence, from which we can easily optimize an ideal classifier $\Phi^*(x)$ with the derived cross-entropy loss. Here minimizing $L_1'$ denotes training the local models in parallel on client $k$, whereas minimizing $L_2'$ can optimize the objective in the DKD module to teach the global model learning from well-trained local models.

Motivated by the above, we propose a federated learning framework equipped with DKD, named FedDKD, composed of two stages per round. Each client is trained locally with ground truth labels on private data (min $L_1'$). Then the global model learns from local models in the DKD
module, of which the corresponding objective is formulated as follows:

$$\sum_{k \in C} q_k \sum_{(x, y) \in D_k} L_{CE}(\Phi(x; w_k), \Phi(x; w)),$$

(7)

where $\sum_{k \in C} q_k = 1$, and $L_{CE}(\cdot, \cdot)$ is the cross-entropy loss function. The optimal solution of minimizing (7) provides the center of all local models $\Phi(x; w_k), k = 1, \cdots, K$, in the function space rather than in the parameter space according to the (possibly weighted) Euclidean space.

**Input:** Private datasets $D_k, k = 1, 2, \cdots, K$

**Federated learning parameters:** local epoch $E$, local mini-batch size $B$, a fraction of clients involved in each round $c$, local learning rate $\eta$, the DKD round $T$

**DKD parameters:** DKD step $J$, DKD learning rate $\gamma$, DKD mini-batch size $\tilde{B}$

**Output:** The updated global model $w_T^g$

```plaintext
procedure SERVER :
    1: for Each DKD round $t = 1, \cdots, T$ do
    2:     clients $m \leftarrow \max(c \cdot K, 1)$
    3:     Randomly sample a set of $m$ clients $C_t$
    4:     For each client $k \in C_t$, in parallel do
    5:         $w_k \leftarrow$ Client-LocalTrain($k, w_{t-1}^g$)
    6:     Get the initial global parameter
    7:         $\tilde{w}_{t, 0} = \sum_{k \in C_t} \frac{n_k}{\sum_{k \in C_t} n_k} w_k$
    8:     for DKD step $j \in \{1, \cdots, J\}$ do
    9:         For each client $k \in C_t$, in parallel do
   10:            $\nabla d_{k, j} \leftarrow$ DKDSGD($\tilde{w}_{t, j-1}, w_k, D_k$)
   11:     $\tilde{w}_{t, j} = \tilde{w}_{t, j-1} - \gamma (\sum_{k \in C_t} q_k \nabla d_{k, j})$
   12:     end for
   13:     $w_T^g = \tilde{w}_{t, J}$
   14: end for
   15: return $w_T^g$

Client-LocalTrain($k, w$):
   1: $B \leftarrow$ split $D_k$ into batches of size $B$
   2: for each local epoch $i = 1, 2, \cdots, E$ do
   3:     for batch $b$ in $D_k$ do
   4:         $w \leftarrow$ optimizer($b; w; \eta$)
   5:     end for
   6: end for
   7: return $w$ to server
```

Algorithm 1 FedDKD.

FedDKD Pipeline We summarize the FedDKD training pipeline in Algorithm 1. For an easier understanding of the DKD process, we provide some essential definitions below.

1. A **communication round** includes the clients uploading their parameters or gradients to the server, and then the server distributing the global model to clients.
2. In a **DKD round**, the server distributes the global model to clients, followed by the clients training local models with that on their local datasets and then uploading their parameters or gradients to the server to obtain a new global model. For FedDKD, a DKD round includes the DKD module with $J$ DKD steps.
3. The **DKD step** refers explicitly to the training step on a single minibatch in the DKD module.
4. The **training step** in this paper only represents the training step of the local models on a single minibatch.
5. The **inference step** here denotes the extra cost inference step on a single minibatch, e.g., the inference step on a single minibatch in the DKD module to obtain the teachers’ output or the inference process of the global model or local historical model on a single minibatch in MOON [17].

In each DKD round $t$, we first sample a subset $C_t$ of clients. The local parameters are initialized with $w_{t-1}^g$ (i.e., a global model estimation in the last DKD round), meaning “download the updated global parameters” as in Fig. 3. Then, the clients optimize the local models using local data in parallel for $E$ local epochs in “local training”.$^t$. Afterward, the online clients upload their local models (i.e., “upload local parameters”), the server averages them to obtain the global model (i.e., “parameter average”), and the online clients download the latest global model (i.e., “download averaged parameters” in Fig. 3). In the following sever DKD steps named the DKD module, by minimizing (7), the clients perform local distillation and upload the knowledge to join the distillation gradient average by several communication rounds (i.e., “upload local distillation gradient” and “download updated global parameters”). Finally, the global model is downloaded for the local training of the online clients in the next DKD round $t + 1$ (i.e., “download updated global parameters”).

**DKD module training details** With the empirical distribution of the sampled local dataset $D_k$ on client $k$, we use the SGD to estimate the gradient of the subterm $\frac{1}{n_k} \sum_{(x, y) \in D_k} L_{CE}(\Phi(x; w_k), \Phi(x; w))$. The algorithm to calculate the gradient on each client is DKDSGD (Algorithm 2). The DKDSGD calculates the local distillation gradient on the minibatch drawn from the local distribution and uploads the gradients to the server in each DKD step $j$. Then, the server collects all local distillation gradients $\nabla d_{k,j}, k \in C_t$ from local clients and averages them to obtain the gradient of (7) concerning $\tilde{w}_{t, j}$. After acquiring the gradient for (7), the global model updates with the DKD learning rate $\gamma$ and distributes the new global parameters to
the local clients. In FedDKD (Algorithm 1), Lines 7 to 11 represent the DKD module.

**Algorithm 2**: DKDSGD. Solver with SGD on each client $k$ in step $j$.

1. Sample a minibatch $B_k$ from training data $D_k$ with batch size $\bar{B}$.
2. Get the logit $\Phi(B_k; w_k)$ of local model on client $k$ with local parameters $w_k$ and the logit $\Phi(B_k; w)$ of the global model with global parameters $w$.
3. $\nabla d_{k,j} = \frac{1}{\bar{B}} \sum_{(x,y) \in B_k} L_{CE}(\Phi(x, w_k), \Phi(x, w))$.
4. return $\nabla d_{k,j}$

**FedDKD Benefits** Minimizing $L'_1$ and $L'_2$ in (5) alternately, FedDKD minimizes the upper bound of the distance between the ground truth map and the global model in the function space. The core of FedDKD is minimizing both terms, whereas most federated learning algorithms only optimize $L'_1$ and then obtain an averaged model in the parameter space. Eliminating the damage caused by the heterogeneity of the local datasets and neural network reparameterization invariance on the map functions by minimizing $L'_2$, the DKD module helps FedDKD search for a better global model in the function space for the current DKD round. Additionally, it sends back better initial parameters for the next DKD round in local training.

### 3.3 Extensions of FedDKD

The FedDKD framework can be a general framework of federated learning because the DKD module can be a plug-in into other federated learning methods.

**FedDKD_MAX** We replace the local training scheme of FedDKD with that of FedMAX [10] to enable similar activation vectors across various clients. In detail, we add a loss term $\frac{1}{\bar{B}} \sum_{i=1}^{N} K L(a_i \| U)$ to the cross-entropy loss in local training on the clients, where $a_i$ is the activation vector of the sample $i$, $\beta$ is a hyperparameter, and $U$ is the uniform distribution over the activation vectors. We call the alternative algorithm FedDKD_MAX.

**FedDKD_BN** We combine FedDKD with FedBN [12] to handle the neural network models with batch normalization (BN) layers and call it FedDKD_BN, where local models have their own BN layers whose parameters remain. Thus, for FedDKD_BN, the objective of the DKD module is revised as follows:

$$
\min_{w' \in \mathbb{R}^d} \sum_{k \in \mathcal{C}} \sum_{(x,y) \in D_k} L_{CE}(\Phi_k(x; w_k), \Phi(x; w', w_k^{BN})),
$$

where $w_k^{BN}$ is the parameter of the BN layers of the local model $k$, and $w'$ represents other parameters. In the DKD module of FedDKD_BN, only the weight $w'$ is updated. Based on FedBN, FedDKD_BN takes weighted average parameters, excluding the BN layers, as the initial global model of the DKD module.

Please refer to Supplementary Materials D and E for more details on these extensions.

### 4 Experiment and results

We perform plentiful numerical experiments on EMNIST/FEMNIST [39, 40], CIFAR-10/CIFAR-100 [41], and a multi-source digits dataset compared with several mainstream methods, including FedAvg [1], FedProx [9], FedMAX [10], MOON [17], and FedBN [12]. An experiment on EMNIST compares several state-of-the-art data-free KD approaches, such as FedGen [19] and FedDistill [18]. We experiment with the following sufficient settings:

- various datasets, such as EMNIST, FEMNIST, CIFAR10, CIFAR100, and multi-source digits datasets;
- various heterogeneities: part of the categories, Dirichlet distribution with different $\alpha$ values, and different data sources;
- different activation ratios of clients: 0.25, 0.5, 0.75, and 1.0;
- different local epochs.

We also experiment with the following sufficient metrics:

- test accuracy,
- costs to reach the target accuracy,
- test accuracy under the same communication cost,
- test accuracy under the same training steps,
- the extra inference steps per client.

In addition, we use different models with or without BN layers.

### 4.1 FEMNIST

In this section, we demonstrate the superiority of FedDKD in communication efficiency and convergence speed when handling heterogeneous datasets. Moreover, we display the power of the DKD module equipped with other state-of-the-art training techniques for the local models.
We compare FedDKD with various baselines in experiments on the FEMNIST dataset divided into heterogeneous local datasets. We randomly choose 6 out of 26 classes to allocate to each agent. The distributions of local datasets are shown in Figure F2. We use 50% of the raw training dataset as the federated learning training dataset and 10% as the validation dataset. The test dataset remains the same as the raw dataset.

We set the number of clients to 20, and the activation ratio of clients per DKD round $c$ is 0.25, 0.5, or 0.75. The number of local training epochs is 5 or 15, and the total DKD rounds $T$ is 660. For FedProx and FedMAX, we perform the optimal search for the best parameters $\mu$ and $\beta$, respectively. For FedDKD and FedDKD_MAX, we set the number of DKD steps $J$ to 10. The DKD learning rate $\gamma$ is initialized as 0.2 with a decay rate of 0.98 per DKD round. We run each experiment with different random seeds three times to obtain credible results. The test accuracy is based on the best model for the validation dataset. Please refer to Supplementary Materials F for more details.

### 4.1.1 Reaching a target quickly and efficiently

Three metrics are considered to reach a target test accuracy, including the DKD rounds, communication rounds, and averaging training steps per activated client. FedDKD requires fewer DKD rounds and training steps in all experimental settings and is efficient in communication when the activation ratio $c$ is 0.5 or 0.75, as the DKD module can estimate the global distribution of the dataset better. For example, if the activation ratio $c = 0.75$ and the local training epoch $E = 5$, FedDKD can achieve a 90.5% test accuracy with only 5% of the DKD rounds and training steps and 50% of the communication rounds compared to those in other mainstream methods, as Table 1 presents. The best results are highlighted in bold. The visualization of Table 1 is shown in Supplementary Materials F. Combining the DKD module with FedMAX, FedDKD_MAX requires fewer training steps and DKD rounds when $c = 0.25$ and $E = 5$.

#### Best performance

After training for 660 DKD rounds, FedDKD and FedDKD_MAX outperform all the other methods by approximately 0.75%, 1.0%, and 1.5% on test accuracy for $c = 0.25$, 0.5, and 0.75, respectively (Table 2). The best results are highlighted in bold. The visualization of Table 2 is shown in Supplementary Materials F. With more clients activated in each communication round, FedDKD can better estimate the global distribution of the datasets and has more advantages than other methods on test accuracy. To compete with other methods fairly in communication cost, we also present the test accuracy when FedDKD and FedDKD_MAX are only trained for 60 DKD rounds. In this setting, all methods communicate for 660 rounds with their local clients, and the communication cost in each round remains the same. However, the average training steps of FedDKD and FedDKD_MAX are fewer than the baselines, and the training cost is approximately 10% of the baselines when $c = 0.5$ and 0.75. The results reveal that FedDKD can achieve better test accuracy with the same training steps and fewer DKD rounds. When the activation ratio $c$ increases, with the same communication cost, FedDKD can achieve better performance with fewer training steps and DKD rounds.

#### Efficient communication

FedDKD also communicates efficiently when the activation ratio $c$ is 0.5 and 0.75 (Fig. 5).

| Methods | $E$ | $c = 0.25$ | $c = 0.5$ | $c = 0.75$ |
|---------|-----|----------|----------|----------|
|         |     | round$^1$ | step$^2$ | comm$^3$ | round$^1$ | step$^2$ | comm$^3$ | round$^1$ | step$^2$ | comm$^3$ |
| FedAvg  | 5   | 330 ± 76  | 81 ± 19  | 330 ± 76 | 245 ± 90  | 60 ± 22  | 245 ± 90 | 283 ± 145 | 69 ± 36  | 283 ± 145 |
|         | 15  | 353 ± 86  | 260 ± 63 | 353 ± 86 | 350 ± 98  | 257 ± 72 | 350 ± 98 | 332 ± 115 | 244 ± 85 | 332 ± 115 |
| FedProx | 5   | 330 ± 76  | 81 ± 19  | 330 ± 76 | 263 ± 112 | 64 ± 28  | 263 ± 112 | 301 ± 145 | 74 ± 36  | 301 ± 145 |
|         | 15  | 425 ± 25  | 313 ± 18 | 425 ± 25 | 339 ± 113 | 249 ± 83 | 339 ± 113 | 339 ± 158 | 249 ± 116 | 339 ± 158 |
| FedMAX  | 5   | 310 ± 101 | 76 ± 25  | 310 ± 101| 258 ± 73  | 63 ± 18  | 258 ± 73 | 276 ± 105 | 68 ± 26  | 276 ± 105 |
|         | 15  | 388 ± 90  | 285 ± 66 | 388 ± 90 | 367 ± 98  | 270 ± 72 | 367 ± 98 | 376 ± 127 | 276 ± 93 | 376 ± 127 |
| FedDKD  | 5   | 98 ± 4    | 25 ± 1   | 1074 ± 40| 18 ± 3    | 5 ± 1    | 198 ± 32 | 13 ± 2    | 3 ± 0    | 139 ± 21  |
|         | 15  | 104 ± 12  | 77 ± 9   | 1140 ± 129| 22 ± 8   | 17 ± 6   | 246 ± 93 | 12 ± 2    | 9 ± 2    | 132 ± 24  |
| FedDKD_MAX | 5 | 96 ± 6   | 24 ± 2  | 1052 ± 70 | 19 ± 3 | 5 ± 1 | 205 ± 36 | 15 ± 3 | 4 ± 1 | 169 ± 29 |
|         | 15  | 151 ± 47  | 113 ± 35 | 1665 ± 519| 23 ± 8 | 17 ± 6 | 257 ± 87 | 13 ± 2 | 9 ± 1   | 139 ± 21 |

$^1$DKD rounds,

$^2$local training step

$^3$communication rounds. If not mentioned, the followings are the same as that
The proposed method achieves the best test accuracy with the same communication rounds and communication costs. For detailed results for $E = 15$, please refer to Supplementary Materials F. With the same communication rounds, FedDKD and FedDKD_MAX only run for $1/11$ of the DKD rounds and approximately $10\%$ of the training steps of the baselines. Thus, the proposed methods may be weaker than other methods when the communication rounds are much fewer.

**Efficient training** The proposed method requires a few more training steps per DKD round in the DKD module. However, the improvement is not due to more training steps, and FedDKD is training-efficient. With the same training steps, FedDKD and FedDKD_MAX display a considerable improvement in test accuracy compared to baselines (Fig. 6). A rational explanation is that many unnecessary training steps occur in the baselines, and more communication is critical in pursuing better performance. Please refer to Supplementary Materials F for the results of $E = 15$ and $E = 5, c = 0.25$.

### 4.2 CIFAR-10/100

In this experiment, we consider a more challenging task of the datasets CIFAR-10 and CIFAR-100 [41] to illustrate

---

### Table 2 Test accuracy on the FEMNIST dataset

| Methods | $E$ | $c = 0.25$ | Round | Test accuracy | $c = 0.5$ | Round | Test accuracy | $c = 0.75$ | Round |
|---------|-----|------------|-------|---------------|------------|-------|---------------|------------|-------|
| FedAvg  | 5   | 0.9156 ± 0.0026 | 521 ± 20 | 0.9128 ± 0.0027 | 523 ± 16 | 0.9126 ± 0.0028 | 519 ± 16 |
|         | 15  | 0.9114 ± 0.0038 | 586 ± 43 | 0.9069 ± 0.0041 | 607 ± 16 | 0.9077 ± 0.0034 | 589 ± 9  |
| FedProx | 5   | 0.9134 ± 0.0045 | 459 ± 52 | 0.9127 ± 0.0043 | 513 ± 17 | 0.9113 ± 0.0029 | 506 ± 37 |
|         | 15  | 0.9062 ± 0.0036 | 527 ± 1  | 0.9088 ± 0.0032 | 639 ± 16 | 0.9067 ± 0.0053 | 600 ± 43 |
| FedMAX  | 5   | 0.9158 ± 0.0050 | 526 ± 14 | 0.9152 ± 0.0042 | 518 ± 29 | 0.9137 ± 0.0036 | 516 ± 7  |
|         | 15  | 0.9098 ± 0.0049 | 562 ± 29 | 0.9073 ± 0.0045 | 534 ± 28 | 0.9059 ± 0.0042 | 570 ± 61 |
| FedDKD  | 5   | 0.9245 ± 0.0015 | 460 ± 57 | 0.9246 ± 0.0023 | 271 ± 220| 0.9273 ± 0.0014 | 113 ± 33 |
|         | 15  | 0.9189 ± 0.0001 | 613 ± 56 | 0.9200 ± 0.0012 | 261 ± 220| 0.9227 ± 0.0020 | 76 ± 25  |
| FedDKD_MAX | 5   | **0.9251 ± 0.0012** | 480 ± 40 | **0.9259 ± 0.0011** | 238 ± 184| **0.9275 ± 0.0017** | 93 ± 27  |
|         | 15  | 0.9174 ± 0.0018 | 560 ± 31 | 0.9194 ± 0.0005 | 109 ± 17 | 0.9211 ± 0.0018 | 67 ± 20  |
| FedDKD@60 | 5   | 0.8904 ± 0.0078 | 53 ± 5  | 0.9202 ± 0.0014 | 51 ± 10  | 0.9257 ± 0.0006 | 53 ± 5   |
|         | 15  | 0.8875 ± 0.0116 | 55 ± 4  | 0.9183 ± 0.0012 | 52 ± 9   | 0.9222 ± 0.0026 | 40 ± 3   |
| FedDKD_MAX@60$^\dagger$ | 5   | 0.8915 ± 0.0140 | 50 ± 9  | 0.9230 ± 0.0024 | 51 ± 10  | **0.9275 ± 0.0010** | 58 ± 1   |
|         | 15  | 0.8795 ± 0.0089 | 52 ± 5  | 0.9165 ± 0.0024 | 52 ± 8   | 0.9213 ± 0.0020 | 50 ± 7   |

$^\dagger$Take 60 DKD rounds

---

![Fig. 5](image-url) Test accuracy under the same communication cost for $c = 0.5$ (left) and $c = 0.75$ (right), where $E = 5$
that the proposed method is efficient and outperforms the baseline methods in much more complex tasks.

Following the previous work [13], we divide CIFAR-10 into $K$ clients by sampling $p_r \sim \text{Dir}_K(\alpha)$ from a Dirichlet distribution and allocating a $p_{r,k}$ proportion of the training instances of class $r$ to the local client $k$. In the experiments, $\alpha = 0.1$, and the local datasets are highly heterogeneous. We use the same neural network model (i.e., VGG-9) without BN layers, as in [13]. We set 10% of the raw training set as the validation set and the others as the training set.

In the CIFAR-10/100 experiments, we set the number of clients to 16, and the online client fraction $c$ is 1.0 (i.e., all clients will be involved in the local update per DKD round). To achieve a trade-off between performance and communication cost, we set DKD step $J = 3$. The DKD learning rate $\gamma$ is 0.08 with a decay rate of 0.99 per DKD round for CIFAR-10 and $\gamma = 0.06$ for CIFAR-100. The test accuracy and corresponding DKD and communication rounds are based on the best model for the validation set. For FedProx, we search the hyperparameter $\mu \in \{0.001, 0.01, 0.1, 1.0, 5.0\}$ and determine the best value of 0.1. The output dimension of the projection head is 512. Please refer to Supplementary Materials G for the details of the experimental settings.

In CIFAR-10 (Table 3), with 350 DKD rounds, the proposed FedDKD improves the test performance by almost 5% of FedAvg and FedProx at the expense of similar training costs and four times the communication cost. With a similar communication cost and fewer training steps (i.e., 24.4% of the training steps of the baselines), FedDKD achieves almost 2% improvement in the test accuracy in 87 DKD rounds. To reach the same test accuracy target (i.e., 73%) with less communication cost, FedDKD requires only approximately 20% of the DKD rounds and 20% of the training steps of the FedAvg and FedProx. The best results in Table 3 are highlighted in bold. The visualization of Table 3 is shown in Supplementary Materials G.

Compared to MOON in CIFAR-10, FedDKD outperforms MOON by approximately 3.4% in test accuracy, and can achieve better accuracy with less DKD rounds, training steps, and inference steps with 87 DKD rounds. Herein, the additional inference steps in FedDKD are caused by knowledge distillation of the global model, whose number is equal to the additional communication rounds. In MOON,

![Graph](image)

**Fig. 6** Test accuracy under the same training steps for $c = 0.5$ (left) and $c = 0.75$ (right), where $E = 5$

| Methods          | Test accuracy | Test accuracy $\geq$ 73% |
|------------------|---------------|---------------------------|
|                  | Test accuracy | Round | Comm round | Train step | Infer step | Round | Comm round | Train step | Infer step |
| FedAvg           | 0.7515 ± 0.0131 | 346 ± 4 | 346 ± 4 | 153.7 ± 1.7 | 0 | 263 ± 46 | 263 ± 46 | 116.7 ± 20.3 | 0 |
| FedProx          | 0.7497 ± 0.0143 | 345 ± 1 | 345 ± 1 | 153.2 ± 0.6 | 0 | 250 ± 65 | 250 ± 65 | 110.8 ± 28.8 | 0 |
| MOON             | 0.7739 ± 0.0190 | 328 ± 10 | 328 ± 10 | 145.6 ± 5.2 | 291.1 ± 10.5 | 131 ± 28 | 131 ± 28 | 58.0 ± 15.0 | 116.0 ± 30.0 |
| FedDKD@87†      | 0.7760 ± 0.0062 | 85 ± 2 | 340 ± 7 | 37.7 ± 1.1 | 0.3 ± 0.0 | 49 ± 2 | 197 ± 10 | 22.0 ± 1.1 | 0.1 ± 0.0 |
| FedDKD          | **0.8076 ± 0.0043** | 294 ± 35 | 1175 ± 138 | 131.2 ± 15.5 | 0.9 ± 0.1 | **49 ± 2** | **197 ± 10** | **22.0 ± 1.1** | **0.1 ± 0.0** |

†Take 87 DKD rounds
the additional inference steps are caused by the calculation of the model-contrastive loss in the training process, which is twice the local training steps.

To determine the stability of FedDKD on data heterogeneity, we experiment with various data heterogeneity $\alpha$ on CIFAR-10. According to Fig. 2, when local datasets are more heterogeneously distributed on clients, the averaged global model in the function space has more advantages than the averaged model in the parameter space, which is consistent with our common intuition. Herein, we compare the function-averaged method FedDKD with other mainstream parameter-averaged federated learning methods to explore the reaction of each framework to different data heterogeneity in Table 4. The best results are highlighted in bold. The visualization of Table 4 is shown in Supplementary Materials G. The function-averaged method FedDKD is significantly efficient in alleviating the performance decay produced by data heterogeneity. With the data heterogeneity increasing, FedDKD outperforms other parameter-averaged methods by approximately 3% and 6%, respectively, when $\alpha$ is 0.1 and 0.05. The superiority of our method becomes more apparent as the data heterogeneity increases and the gap between the parameter-averaged model and function-averaged model widens.

For the more challenging dataset CIFAR-100, which has 100 categories, local teachers have low test accuracy (less than 40%) in the first 30 DKD rounds, and the knowledge is too incomplete to teach the student model. Thus, the DKD module is not executed for this experiment’s first 30 DKD rounds. FedDKD improves the global model by approximately 2% regarding test accuracy (Table 5). It only needs 1/3 of the DKD rounds and training steps in other methods to reach the same test accuracy target (i.e., 51.4%) under the same communications, which illustrates the training efficiency of the proposed method. For a fair comparison, we also run FedDKD for 110 DKD rounds (i.e., FedDKD@110), where the number of communication rounds is 350. In this case, FedDKD still slightly exceeds the baseline in test accuracy. However, FedDKD only trains approximately 1/3 of the steps. The best result in Table 5 is highlighted in bold.

The experiments on CIFAR-10/100 illustrate that the proposed method is efficient in much more complex tasks with equal DKD rounds and training steps. With similar communication costs and fewer training steps, FedDKD exceeds the baselines on test accuracy. The experiments imply that local training is often superfluous for heterogeneous datasets and that more communication is the key to higher performance.

### 4.3 Comparison with knowledge distillation algorithms

This subsection illustrates that as a data-free method, FedDKD performs better than other data-free FL algorithms equipped with KD, such as FedGen and FedDistill.

In FedGen, the server trains a generator to ensemble the client information in a data-free manner and then broadcasts it to users, regulating local training using the learned knowledge as an inductive bias. Additionally, FedDistill only collects and returns the average logits of the client models. In previous research [19], they also share the parameters of the client models as FedAvg for FedDistill, called FedDistill$^+$, to achieve a fair comparison.

The experiments on the EMNIST dataset follow the work in [19]. There are 20 clients with an active ratio of 50%. The local epoch $E = 20$. We use only 10% of the training dataset and all test datasets. To obtain an extremely heterogeneous dataset, we divide the training dataset according to the Dirichlet distribution with $\alpha = 0.05$ or 0.1. We maintain the hyperparameters for the baseline as those in FedGen [19]. For FedDKD, we set the DKD learning rate $\gamma = 0.40$ with a decay rate of 0.99 per DKD round, and the DKD steps $J = 3$. We run each experiment 10 times.

As Table 6 shows, FedDKD achieves the best test accuracy and outperforms the other methods by approximately 4.2% when $\alpha = 0.05$ and by 1.7% when $\alpha = 0.1$. The best results in Table 6 are highlighted in bold. The visualization of Table 6 is shown in Supplementary Materials H. In addition, FedDKD does not require a well-designed and well-trained extra generator as in FedGEN but achieves superb performance when the dataset is highly heterogeneous.

### 4.4 Architecture with the batch normalization layer

In this experiment, we consider the architecture with the BN layer and illustrate that FedDKD has excellent scalability for model structures with the BN layer.

| Methods    | $\alpha = 0.05$ | $\alpha = 0.1$ | $\alpha = 0.5$ |
|------------|----------------|----------------|----------------|
| FedAvg     | 0.5619 ± 0.0174 | 0.7515 ± 0.0131 | 0.8754 ± 0.0051 |
| FedProx    | 0.5724 ± 0.0208 | 0.7497 ± 0.0143 | 0.8750 ± 0.0035 |
| MOON       | 0.6159 ± 0.0268 | 0.7739 ± 0.0190 | 0.8749 ± 0.0062 |
| FedDKD     | 0.6760 ± 0.0094 | 0.8044 ± 0.0081 | 0.8775 ± 0.0029 |
Table 5 Performance overview of CIFAR-100

| Methods       | Test accuracy | Round | Comm round | Train step | Infer step |
|---------------|---------------|-------|------------|------------|------------|
| FedAvg        | 0.5188 ± 0.0041 | 339 ± 3 | 339 ± 3   | 150.4 ± 1.3 | 0          |
| FedProx       | 0.5227 ± 0.0067 | 324 ± 5 | 324 ± 5   | 143.9 ± 2.4 | 0          |
| MOON          | 0.4983 ± 0.0023 | 328 ± 7 | 328 ± 7   | 145.0 ± 3.6 | 290.0 ± 7.1|
| FedDKD@110†   | 0.5241 ± 0.0109 | 105 ± 4 | 330 ± 14  | 39.4 ± 7.0 | 0.330 ± 0.0 |
| FedDKD        | **0.5424 ± 0.0130** | 274 ± 48 | 1007 ± 193 | 115.1 ± 20.5 | 1.007 ± 0.2 |

†Take 110 DKD rounds

Most models remove the BN layer in federated learning to average the model better. However, FedBN introduces the BN layer to handle heterogeneous datasets when the local datasets are collected from different sources. This subsection primarily demonstrates the power of the DKD module equipped with other state-of-the-art methods.

Specifically, following the work [12], we use five different digit datasets: SVHN [42], USPS [43], SynthDigits [44], MNIST-M [44], and MNIST [45], with 10% of each dataset allocated to each client with 743 training samples for each local model. We use the same neural network model as in FedBN, with three convolutional layers with BN layers and three fully connected layers with BN layers. Please refer to Supplementary Materials I for the details of this experiment.

Table 7 illustrates that the FedDKD_BN achieves the best test accuracy and outperforms FedAvg by 3.21%, FedProx by 3.14%, FedBN by 0.31%, and FedDKD by 0.58%. The best results in Table 7 are highlighted in bold. Furthermore, FedDKD is slightly weaker than FedBN in test accuracy. However, for FedBN, every local model maintains its BN layer, which means it has $K$ models for $K$ clients, sharing most parameters and maintaining the respective batch normalization layer. In contrast, FedDKD has only a single global model.

The DKD module can work with the model with BN layers. Moreover, combining the technique for FedBN can improve performance. This finding also implies that the DKD module can be a general technique combined with other training methods for local models to improve performance.

Table 6 Test accuracy compared with knowledge distillation algorithms on the EMNIST dataset

| Methods       | $\alpha = 0.05$ | $\alpha = 0.1$ |
|---------------|----------------|----------------|
| FedAvg        | 64.76 ± 2.11   | 69.07 ± 1.71   |
| FedProx       | 63.90 ± 2.13   | 68.49 ± 1.74   |
| FedDistill+   | 63.18 ± 2.31   | 69.20 ± 1.44   |
| FedGEN        | 68.56 ± 1.82   | 72.06 ± 1.61   |
| FedDKD        | **72.77 ± 1.38** | **73.75 ± 1.60** |

4.5 Ablation experiments of DKD step $J$

The DKD step $J$ is an essential hyperparameter for our FedDKD algorithm, which balances the trade-off between test accuracy and communication cost. In this subsection, we indicate the effect of the DKD step on the performance of the FedDKD algorithm.

Particularly, when $J = 0$, the FedDKD degrades into the FedAvg algorithm. In this experiment, we show the influence of the DKD step $J$ for the FEMNIST and CIFAR-10 datasets. In this experiment, Fig. 7 shows the test accuracy of different algorithms and FedDKD with different DKD steps in 100 DKD rounds.

More DKD steps in each DKD round improve the test accuracy when $J$ is not too large. However, there is no apparent benefit to increasing the DKD steps from 10 to 15 for the FEMNIST dataset. Additionally, more DKD steps indicate more communication costs in each DKD round. To be exact, the communication cost of FedDKD with $J$ DKD steps is $J + 1$ times that of baselines. To reach a trade-off between test accuracy and total communication cost, we choose $J = 10$ for the FEMNIST experiments and $J = 3$ for the CIFAR-10 experiments. The experiments also reveal that communication between the server and clients has a more critical impact than relatively redundant local training epochs.

5 Discussion

This section introduces the second-order Taylor approximation of the optimized function (6). In this perspective, we discuss the relation of FedDKD to the parameter-averaged model. Motivated by that, we further discuss the approach to extend FedDKD to a second-order method as future work.

5.1 Relation to the parameter-averaged model

In this subsection, we discuss the relation to the parameter-averaged model of the optimization objective function of the DKD module and demonstrate why FedDKD performs
Table 7  The results for the multi-source digits dataset in 100 DKD rounds

| Methods       | FedAvg   | FedProx  | FedBN    | FedDKD   | FedDKD_BN |
|---------------|----------|----------|----------|----------|-----------|
| Best test accuracy | 82.57% ± 0.23% | 82.64% ± 0.42% | 85.47% ± 0.22% | 85.20% ± 0.21% | **85.78% ± 0.22%** |
| Final test accuracy  | 82.44% ± 0.22% | 82.36% ± 0.24% | 85.39% ± 0.30% | 85.02% ± 0.25% | **85.67% ± 0.23%** |

similarly to the parameter averaging method when the local datasets are homogeneous.

Taking the weighted average of the local models in the parameter space (i.e., \( \hat{w} \approx \sum_{k=1}^{K} q_k w_k \)) as the approximation of \( \hat{w} \) in (6) is a mainstream method in many federate learning algorithms. We analyze its rationality in the homogeneous dataset in terms of the DKD module and demonstrate why it does not work for a heterogeneous dataset.

A second-order Taylor approximation of the optimized function in (6) on a fixed point \( w^* \) is defined as follows:

\[
\sum_{k=1}^{K} q_k \int_{D_k} d\text{iv}(\Phi(x; w_k), \Phi(x; w)) P_k(dx)
\]

\[
= \frac{1}{2} \sum_{k=1}^{K} q_k \langle w_k - w^* | H_k(w^*) \rangle w_k - w^* \rangle + O(\|w_k - w^*\|^3) + O(\|w - w^*\|^3),
\]

(8)

where \( H_k(w^*) = \int_{D_k} \frac{\partial \Phi(x; w^*)}{\partial w}^\top H_\Phi(x; w^*) \frac{\partial \Phi(x; w^*)}{\partial w} P_k(dx) \) and \( H_\Phi(x; w^*) = \frac{\partial^2 \text{div}(u, v)}{\partial u \partial v} |_{u=v=\Phi(x; w^*)} \). The following proposition indicates the effectiveness of the weighted average model when datasets are homogeneous.

**Proposition 2** Assume that \( \text{div}(u, v) \) is differentiable for any possible \( u \) and \( v \), and the datasets \( D_k, k = 1, 2, \cdots, K \) on the clients are homogeneous (i.e., \( H_k(w^*) = H(w^*), \forall k = 1, \cdots, K \)), then \( \sum_{k=1}^{K} q_k w_k \) is one of the solutions of the following

\[
\text{arg min}_w \frac{1}{2} \sum_{k=1}^{K} q_k \langle w_k - w | H_k(w^*) \rangle w_k - w^* \rangle.
\]

Moreover, Corollary 1 is a linear form of Proposition 2. Detailed proof of Proposition 2 and Corollary 1 is provided in Supplementary Materials J.

**Corollary 1** If \( \Phi(x; w) = \Phi(x; (A, b)) = Ax + b \), where \( x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m \), \( \text{div}(u, v) = \|u - v\|^2_2 \), and \( D_k, k = 1, \cdots, K \) are all homogeneous datasets, then one of the solutions of (6) is that \( \hat{w} = \sum_{k=1}^{K} q_k w_k \), which is the weighted average of the local models.

For a heterogeneous dataset, a more general case in federated learning, there is a high probability that Proposition 2 and Corollary 1 do not hold because \( H_k(w^*), k = 1, 2, \cdots, K \) may be very different. As a result, for heterogeneous datasets, only minimizing \( L_1 \) in (3) and then taking the weighted average of local models cannot guarantee the optimality of the global model. Thus, the DKD module is essential to minimize \( L_2 \) in (3) to search for a better global model in the function space under data heterogeneity situations.

Fig. 7 Test accuracy in 100 DKD rounds for FEMNIST (left) and CIFAR-10 (right) for baselines and FedDKD with different DKD step \( J \).
5.2 Extension to second-order methods

In this subsection, we show how to extend the FedDKD to a second-order method when training the DKD module, which serves as an extension for future work.

In this section, we assume \( q_k = 1/K \), \( k = 1, 2, \ldots, K \). Given a mini-batch \( B_k \) for the client \( k \), for any rational \( w \), we can find the update of the weights \( d \) for the central server.

\[
\min_d \sum_{k=1}^{K} d \, i \, u(\Phi(x_{B_k}, w + \Delta w_k), \Phi(x_{B_k}, w + d)),
\]

where \( \Delta w_k = w_k - w \). Using the second-order Taylor expansion, we have

\[
\sum_{k=1}^{K} H_k(w)(\Delta w_k - d) = 0
\]

where \( H_k(w) = \left( \frac{\partial \Phi(x_{B_k}, w)}{\partial w} \right)^T H_{B_k} \frac{\partial \Phi(x_{B_k}, w)}{\partial w} \) and \( z_{B_k} = \Phi(x_{B_k}, w) \), \( H_{B_k} = \frac{\partial^2 \Phi(x_{B_k}, w)}{\partial w^2} \). Hence, the parameter-averaged method can be viewed as a particular case that simply treats \( H_k(w) = I \) in (10).

If \( H_k(w) \) is computable and easy to store, FedDKD can be extended to utilize second-order optimization methods, which may further reduce the DKD steps per DKD round. However, the computation and storage of the second-order Taylor expansion is a primary challenge in many fields, such as natural gradient, as the neural network parameters are often in the millions. Combining FedDKD with the natural gradient solution method is one of our future research directions.

6 Conclusion

This paper proposes a novel federated learning framework with decentralized knowledge distillation called FedDKD. Directly averaging the local models is not inherently the best method for the permutation invariance of neural networks and the heterogeneity of local datasets. FedDKD utilizes decentralized knowledge distillation to learn what local models learn in the function space rather than the parameter space for the data heterogeneity situations. Experiments on various datasets demonstrate that FedDKD can achieve superb test accuracy, more efficient communication, and faster convergence speeds in some highly heterogeneous datasets.

In practice, FedDKD requires that the clients continuously stay online during \( J + 1 \) communication rounds, which is easy to implement when \( J \) is not too large and does not increase privacy risks compared to universal algorithms. In FedDKD, excessive local training is generally redundant, whereas communication is essential because it helps approach the average in the function space rather than the parameter space for heterogeneous datasets.

In future work, we will explore how to combine efficient second-order algorithms to accelerate the convergence of the distillation stage and further reduce the communication costs of the DKD module.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10489-022-04431-1.

Acknowledgments This work was supported by the National Key R&D Program of China (No. 2018AA0100303); the National Natural Science Foundation of China (No. 62072111); the Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and the ZHANGJIANG LAB; and the Technology Commission of Shanghai Municipality (No. 19JC1420101).

Funding This work was supported by the National Key R&D Program of China (No. 2018AA0100303); the National Natural Science Foundation of China (No. 62072111); the Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and the ZHANGJIANG LAB; and the Technology Commission of Shanghai Municipality (No. 19JC1420101).

Data Availability The additional datasets generated during and analyzed during the current study are available from the corresponding author upon reasonable request.

Code Availability The source code will be available on GitHub.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. McMahan HB, Moore E, Ramage D, Hanson S, Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th international conference on artificial intelligence and statistics (AISTATS)
2. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: Challenges, methods, and future directions. IEEE Signal Proc Mag 37(3):50–60. https://doi.org/10.1109/mssp.2020.2975749
3. Kairouz P, McMahan H, Avent B, Bennis A, Bhatt S, Bhagoji A, Bonawitz K, Charles Z, Cormode G, Cummings R, D’Oliveira R, Eichner H, El Rouayheb S, Evans D, Gardner J, Garrett Z, Gascón A, Ghazi B, Gibbons P, Zhao S (2021) Advances and Open Problems in Federated Learning. https://doi.org/10.1561/9781680837896
4. Imteaj A, Thakker U, Wang S, Li J, Amini MH (2022) A survey on federated learning for resource-constrained iot devices. IEEE Internet of Things Journal 9(1):1–24. https://doi.org/10.1109/JIOT.2021.3095077
5. Wahab OA, Mourad A, Otrok H, Taleb T (2021) Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. IEEE Communications Surveys & Tutorials 23(2):1342–1397. https://doi.org/10.1109/COMST.2021.3058573
6. Su Z, Wang Y, Luan TH, Zhang N, Li F, Chen T, Cao H (2022) Secure and efficient federated learning for smart grid with edge-cloud collaboration. IEEE Transactions on Industrial Informatics 18(2):1333–1344. https://doi.org/10.1109/TII.2021.3095506
7. Zhang W, Zhou T, Lu Q, Wang X, Zhu C, Sun H, Wang Z, Lo SK, Wang F-Y (2021) Dynamic fusion-based federated learning for covid-19 detection. IEEE Internet of Things Journal PP:1–1. https://doi.org/10.1109/JIOT.2021.3056185
8. Zeng Q, Lv Z, Li C, Shi Y, Lin Z, Liu C, Song G (2022) Fedpros: federated learning for iot perception data prediction. Appl Intell, 1–13. https://doi.org/10.1007/s10489-022-03578-1
9. Li T, Sahu AK, Zaeber M, Sanjabi M, Talwalkar A, Smith V (2020) Federated optimization in heterogeneous networks. Proc Mach Learn Syst 2:429–450
10. Chen W, Bhardwaj K, Marculescu R (2021) Fedmax: Mitigating activation divergence for accurate and communication-efficient federated learning. In: Hutter F, Kersting K, Lijffijt J, Valera I (eds) Machine learning and knowledge discovery in databases. Springer, Cham, pp 348–363
11. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Lechevallier Y, Saporta G (eds) Proceedings of COMPSTAT’2010. Physica-Verlag HD, Heidelberg, pp 177–186
12. Li X, Jiang M, Zhang X, Kamp M, Dou Q (2021) Fedbn: Federated learning on non-iid features via local batch normalization. In: International conference on learning representations
13. Wang H, Yurochkin M, Sun Y, Papailiopoulos D, Khazaeni Y (2020) Federated learning with matched averaging. In: International conference on learning representations
14. Zhu H, Xu J, Liu S, Jin Y (2021) Federated learning on non-iid data: a survey. Neurocomputing 465:371–390
15. Lin T, Kong L, Stich SU, Jaggi M (2020) Ensemble distillation for robust model fusion in federated learning. In: Proceedings of the 34th international conference on neural information processing systems
16. Mills J, Hu J, Min G (2022) Multi-task federated learning for personalised deep neural networks in edge computing. IEEE Transactions on Parallel and Distributed Systems 33:630–641
17. Li Q, He B, Song D (2021) Model-contrastive federated learning. In: 2021 IEEE/CVF Conference on computer vision and pattern recognition (CVPR). IEEE Computer Society, Los Alamitos, CA, USA, pp 10708–10717, https://doi.org/10.1109/CVPR46437.2021.01057
18. Seoa H, Parkb J, Ohc S, Bennisd M, Kinc S-L. (2022) Federated Knowledge Distillation, pp 457–485. https://doi.org/10.1007/978-1-07166559-019
19. Zhu Z, Hong J, Zhou J (2021) Data-free knowledge distillation for heterogeneous federated learning. In: International conference on machine learning, PMLR, pp 12878–12889
20. Yim J, Joo D, Bae J, Kim J (2017) A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR), pp 7130–7138. https://doi.org/10.1109/CVPR.2017.754
21. Romero A, Kahou SE, Montréal P, Bengio Y, Montréal YD, Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y (2015) Fitnets: Hints for thin deep nets. In: International conference on learning representations
22. Meng Z, Li J, Gong Y, Jiang B-H (2018) Adversarial teacher-student learning for unsupervised domain adaptation. In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 5949–5953
23. Chen G, Choi W, Yu X, Han TX, Chandraker M (2017) Learning efficient object detection models with knowledge distillation. In: NIPS
24. Tan S, Caruana R, Hooker G, Lou Y (2018) Distill-and-compare: Auditing black-box models using transparent model distillation. Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society.
25. Goldblum M, Fowl L, Feizi S, Goldstein T (2020) Adversarially robust distillation. In: AAAI
26. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning 2011. http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf
27. Chelaramani S, Gupta M, Agarwal V, Gupta P, Habash R (2021) Multi-task knowledge distillation for eye disease prediction. In: 2021 IEEE Winter conference on applications of computer vision (WACV), pp 3982–3992. https://doi.org/10.1109/WACV48630.2021.00403
28. Li Z, Hu C, Guo X, Chen J, Qin W, Zhang R (2022) An unsupervised multiple-task and multiple-teacher model for cross-lingual named entity recognition. In: Proceedings of the 60th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Dublin, Ireland, pp 170–179. https://doi.org/10.18653/v1/2022.acl-long.14, https://aclanthology.org/2022.acl-long.14
29. Vongkulbhisal J, Vinayavekhin P, Scaranella MV (2019) Unifying heterogeneous classifiers with distillation. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3170–3179
30. You S, Xu C, Xu C, Tao D (2017) Learning from multiple teacher networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining
31. Liu Y, Zhang W, Wang J (2020) Adaptive multi-teacher multi-level knowledge distillation. Neurocomputing 415:106–113
32. Chen Y, Ning Y, Sławski M, Rangwala H (2020) Asynchronous online federated learning for edge devices with non-iid data. 2020 IEEE International Conference on Big Data (Big Data), 15–24
33. Oh S, Park J, Jeong E, Kim H, Kim SL (2020) Mix2IIFD: Downlink federated learning after uplink federated distillation with two-way mixup. IEEE Commun Lett (PP):1–1
34. Sui D, Chen Y, Zhao J, Jia Y, Sun W (2020) Feded: Federated learning via ensemble distillation for medical relation extraction. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)
35. Duan M, Liu D, Chen X, Tan Y, Ren J, Qiao L, Liang L (2019) Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications, pp 246–254. https://doi.org/10.1109/ICCD46524.2019.00038
36. Amari S-i, Nagaoka H (2007) Methods of information geometry. Translations of Mathematical Monographs, 25–50. https://doi.org/10.1090/mmono/191/02
37. Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Statist 22(1):79–86
38. Amari S-i, Nagaoka H (2007) Methods of information geometry. Translations of Mathematical Monographs, 25–50. https://doi.org/10.1090/mmono/191/02
39. Amari S-i, Nagaoka H (2007) Methods of information geometry. Translations of Mathematical Monographs, 25–50. https://doi.org/10.1090/mmono/191/02
40. Tsymbakov AB (2009) Introduction to Nonparametric Estimation. Springer series in statistics. Springer, Dordrecht. https://doi.org/10.1007/978-1-84628-137-9
41. Cohen G, Afshar S, Tapson J, van Schaik A (2017) Emmist: Extending mnist to handwritten letters. In: 2017 International joint conference on neural networks (IJCNN), pp 2921–2926. https://doi.org/10.1109/IJCNN.2017.7966217
42. Caldas S, Dutdu SMK, Wu P, Li T, Konečný J, McMahan HB, Smith V, Talwalkar A (2018) Leaf: A benchmark for federated settings. arXiv preprint arXiv:1812.01097 Presented at the 2nd International Workshop on Federated Learning for Data Privacy and Confidentiality (in conjunction with NeurIPS 2019)
41. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Technical Report 0. University of Toronto, Toronto, Ontario
42. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning 2011. http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf
43. Hull JJ (1994) A database for handwritten text recognition research. IEEE Trans Pattern Anal Mach Intell 16(5):550–554. https://doi.org/10.1109/34.291440
44. Ganin Y, Lempitsky V (2015) Unsupervised domain adaptation by backpropagation. In: International conference on machine learning, PMLR, pp 1180–1189
45. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Xinjia Li is currently pursuing the Ph.D. degree in applied mathematics at Fudan University, Shanghai, China. His research interests include machine learning, federated learning, signal processing, complex networks and computer vision.

Boyu Chen is pursuing the Ph.D. degree in applied mathematics at Fudan University, Shanghai, China. His research interests include machine learning, Natural Gradient and Meta-learning.

Wenlian Lu received the B.S. degree in mathematics and the Ph.D. degree in applied mathematics from Fudan University, Shanghai, China, in 2000 and 2005, respectively. He was a Post-Doctoral Fellow with the Max Planck Institute for Mathematics in the Science, Leipzig, Germany, from 2005 to 2007, and a Marie-Curie International Incoming Research Fellow with the Department of Computer Sciences, University of Warwick, Coventry, U.K., from 2012 to 2014. He is currently a Professor with the School of Mathematical Sciences and the Institute for Science and Technology of Brain-Inspired AI, Fudan University. His current research interests include neural networks, cybersecurity dynamics, computational systems biology, nonlinear dynamical systems, and complex systems. He served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2013 to 2019 and Neurocomputing from 2010 to 2015.