Citation: He Y, Pan W. Non-IID data based federated learning using mutual knowledge distillation and generative adversarial network. *Journal of Harbin Institute of Technology (New Series)*. DOI: 10.11916/j.issn.1005-9113.25011

Non-IID Data Based Federated Learning Using Mutual Knowledge Distillation and Generative Adversarial Network

Yang He[1] and Weimin Peng[2]*

(*1. School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310000, China;*
2. Key Laboratory of Discrete Industrial Internet of Things, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310000, China)

Abstract: User heterogeneity in federated learning (FL) necessitates the re-optimization of local models, results in the loss of global knowledge, and leads to slow convergence and degraded performance. When dealing with heterogeneous clients in FL, knowledge distillation (KD) is a standard approach to increasing efficiency and improving generalization. However, KD relies on proxy datasets, and the underutilization of client knowledge in guiding local model learning has a negative impact on the quality of the aggregation model. Regarding these, a new FL method is proposed based on generative adversarial network (GAN) and KD, which has two training stages. At the first stage, client collaboration pre-trains a GAN to generate secondary datasets, overcoming the limitations of proxy datasets. In the second stage, the mutual KD process is implemented through the dynamic adjustment of the weights of client models to tackle the underutilization of integrated client knowledge. In the training phase, the pre-trained generator after fine-tuning can transfer knowledge from multiple local models to a global model to enhance the efficiency of KD. On the introduced benchmark datasets, the experimental results show that the proposed FL method needs fewer communication rounds and reflects better generalization than the state-of-the-art FL methods.

Keywords: federated learning; non-independently and identically distributed; knowledge distillation; generative adversarial network

CLC number: TP391.9 **Document code**: A **Article ID**: 1005-9113(2025)00-0000-14

0 Introduction

According to the General Data Protection Regulation (GDPR), data from diverse sources cannot be transmitted to a centralized server. This leads to the creation of a distributed database made up of several data islands. Federated learning (FL) is developed to tackle the data island issue, which allows clients to co-train a generalized and robust model without centralizing their data. Federated averaging (FedAvg)[1] is a well-established FL algorithm. In each round of FedAvg training, the central server distributes the global model weights to a subset of active clients. Each client then trains the model on its local data. Afterward, each active client sends the new model weights back to the central server for the calculation of the updated global model. The central server then sends the new global model to a re-selected group of active clients for the next round of FedAvg training. This cycle repeats until the global model converges.

In the FL training process, a major challenge is brought by the heterogeneously distributed client data. In actual scenarios, the local data of each client may follow a non-independent and identically distributed (non-IID) pattern, which differs from the global data in data distribution. This heterogeneity not only poses challenges to theoretical analysis[2-3], but also degrades model performance and impedes convergence[4-5]. As clients train on their biased local data (e.g., non-IID), their local models gradually lose global

Received 2025-03-06.
This work was supported by National Key R&D Program of China (No. 2024YFB3310800) and Zhejiang Provincial Science and Technology Plan Project (No. 2023C01147).
* Corresponding author: Weimin Peng, Ph.D, Associate Professor. Email: pweimin@163.com.

knowledge, and diverge from the global optimum-a phenomenon known as client drift[6], fundamentally stems from data heterogeneity and poses critical challenges to FL. These challenges include: 1) Local overfitting: The biased local data can lead to the models that excel on client-specific distributions but poorly generalize on the global distribution; 2) Convergence degradation: Divergent local objectives require more communication rounds for global aggregation, which increase training overhead; 3) Negative transfer: Severe drift can cause global models to inherit suboptimal updates and worsen learning performance rather than improve it.

It is a big challenge to address these severe issues of non-IID FL. The existing approaches can be broadly categorized, but each faces critical limitations. With regard to the parameter constraint methods, they face the challenge of stabilizing the local training process[6-8]. SCAFFOLD[6] uses controllable variables to estimate the update directions for both the local and global models. The differences in controllable variables are subsequently used to correct for client drift. FedProx[8] directly limits the locally updated model by incorporating an additional L2 regularization term into the local objective function. In FedDistill[9], clients share the model's output parameters (logits) instead of the model parameters (weights or gradients) to reduce communication costs. Although they mitigate client drift to some extent, they primarily operate in the parameter space and often fail to fully leverage the rich and diverse knowledge embedded in the predictions of local models. The local models are trained on distinct data distributions. Their aggregation (simple averaging) is suboptimal under high heterogeneity.

With respect to knowledge distillation (KD) methods, this approach improves the efficiency of model aggregation[10-11] by leveraging effective KD techniques[12-13]. Here, KD leverages unlabeled datasets as proxies to mitigate the model drift problem caused by user heterogeneity. For example, FedMD[12] uses local models to average the prediction results of public datasets, generates new training labels, and then uses these labels to further train local models. After that, FedMD achieves knowledge transfer without sharing local data. For FedDF[13], an unlabeled dataset on the server is used to combine knowledge from all received local models and enhances the global model with the integrated knowledge. FedDF outperforms the way of simply averaging parameters. However, a fundamental limitation is that FedDF depends on a suitable proxy dataset, which may not be available or representative in many practical FL scenarios. Moreover, by distilling knowledge only into a global model, FedDF fails to fully leverage the collective client knowledge for mutual improvement. FedSKD[14] intelligently selects the most valuable knowledge and transmits it to each client based on task requirements. Using this selective distillation, FedSKD can reduce communication overhead, improve training efficiency, and maintain good model performance. FedAKD[15] adopts the enhanced KD technology to reduce communication overhead in FL and improve the efficiency of human activity recognition in multiple client environments. Meanwhile, FedAKD combines differential privacy technologies to reduce communication requirements between clients and servers and ensure privacy.

Data generation mitigates the impact of data heterogeneity on model performance. FedGen[16] learns a generator with lightweight to generate pseudo features and broadcast them to clients for local training normalization. FedFTG[17] uses a generator with the knowledge learned from local and global models to generate new samples for global model training. However, a key challenge here is the instability of generator training in the early stages of FL. When local models still perform poorly in the early stages, their unreliable predictions can provide misleading signals for generator training, hinder the quality of synthetic data, and potentially harm the overall FL efficiency.

Nevertheless, the existing technologies have some limitations. Specifically, the parameter constraint methods cannot fully exploit predictive knowledge, the KD methods depend on unavailable proxy data, and the data generation methods are vulnerable to unstable early-stage training. Here, a new FL method for non-IID data is developed, which uses mutual KD[18-19] and generative adversarial network (GAN)[20] and is abbreviated as FedMkd-G. This method includes the stages of client-side collaborative training of a generator and generator-based FL, which aims to enhance the generalization performance of standard FL and refines the model aggregation process. In the stage of client-side collaborative training of a generator, a generator

remains on the server side and the client-side data can be used to train a generator. The trained generator is then sent to the server for the next round of training. The next stage of generator-based FL performs mutual KD on the uploaded local models using the auxiliary data produced by the server's generator. The generalization performance of FL can be improved by transferring knowledge from a set of local models to the global model. To prevent the incorporation of misleading knowledge from local models during the mutual learning process, FedMkd-G assesses the accuracy of each local model trained on auxiliary data to adjust the distilled weights of each model accordingly. With the blessing of the effective distillation of the global model during the training process, FedMkd-G iteratively fine-tunes a generator to explore valuable samples, enlarges prediction discrepancies between local and global models, and maximizes the utilization of the knowledge of local models.

This paper has the following contributions:

1) A novel two-stage FL architecture integrating client-collaborative GAN pre-training and dynamic mutual KD. This architecture eliminates the reliance on proxy datasets and adaptively weights local models to mitigate client drift in non-IID scenarios.

2) A dynamic knowledge integration mechanism that prioritizes high-confidence local models during mutual KD. This mechanism reduces negative transfer compared to equal-weight distillation mechanisms and leverages generated auxiliary data to quantify local model reliability.

3) A generator fine-tuning strategy that maximizes discrepancies between local and global models. This generator accelerates convergence and improves accuracy in extreme non-IID scenarios. This co-optimization of the generator and global model addresses the "cold start" problem that exists in data generation methods.

In the following sections, Section 1 presents the proposed FedMkd-G method using mutual KD and GAN. In Section 2 and Section 3, the experimental results of the proposed FedMkd-G method are evaluated and discussed on the benchmark datasets.

1 FedMkd-G Method

1.1 Problem Definition

Given the N_k samples of the k -th client, the local dataset of the k-th client is defined as $D_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k}$. Here, $x_{k,i}$ and $y_{k,i}$ denote the input and label of the i-th element of the k-th client. Given K clients, this paper aims to train a generalized global model ω that is suitable for local and global data distributions and has better generalization ability. Generally, a FL model can be defined as follows:

$$\min_{\omega} \frac{1}{K}\sum_{k=1}^{K} f_k(\omega), \quad f_k(\omega) = \frac{1}{N_k}\sum_{i=1}^{N_k} L(x_{k,i}, y_{k,i}; \omega) \tag{1}$$

here, $f_k(\omega)$ denotes the loss of the k-th client, $L(x_{k,i}, y_{k,i}; \omega)$ is a loss function that measures a training error, the k-th dataset D_k is heterogeneously distributed, and ω denotes a global model. In FL, privacy restrictions prevent the server from directly accessing the local data on client sides. Therefore, the global model ω is sent to a set of randomly selected clients ($S_t = \{c_1, c_2, \cdots, c_k\}$) for optimization (i.e., $\min_{\omega} f_k(\omega)$ ($k \in S_t$)) in the t-th communication round. After that, the server collects local models to update the global model ω by means of gradient averaging.

1.2 FedMkd-G Framework

As shown in Fig. 1, the FedMkd-G framework organized all clients collaboratively to train a conditional GAN (CGAN) model which is then uploaded to the server for additional training. Then the server receives all local models and performs mutual KD on these local models. After that, the server uses these local models to fine-tune a generator for further global model training.

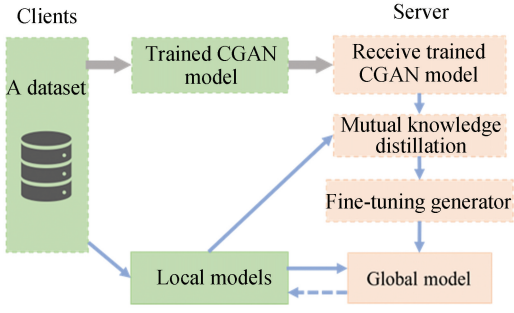

Fig. 1 The proposed FedMkd-G framework

The (cloud) server, which manages the training of local clients (e.g., edge devices), has greater computing and communication resources than clients. Traditionally, the server mainly aggregates model parameters and communicates with clients. These operations are inefficient and cannot fully utilize the server power. In the scenarios with data heterogeneity,

a conventional gradient averaging approach leads to the loss of local model knowledge because of the large differences in local models and degrades the global model performance[21].

In our proposed method, the data-driven strategies and non-IID data are respectively employed on the server and client sides to enhance the generalization performance of FL. The server utilizes the trained generator to generate high-quality synthetic samples for all labeled clients, enables the training of local models on an IID composite dataset, and improves the convergence of the global model. Specifically, the proposed method has two stages of client-side collaborative pre-training of a generator and online FL with auxiliary data. In the first stage, each client collaboratively trains a CGAN with local data and sends the trained CGAN model to the server. In the second stage before aggregation, the server applies mutual KD to enhance each user model's generalization. After aggregation, the server fine-tunes a generator using the global and local models. A generator can produce auxiliary datasets to train and converge the global model.

1.3 Collaborative Training of CGAN

Figs. 2 and 3 show the stages of the collaborative training scheme for client-side generators and the training of the high-performance global model using the generated synthetic data in FL. The corresponding algorithms are defined as Algorithm 1 and Algorithm 2.

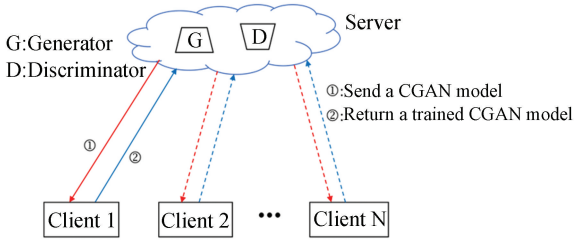

Fig. 2 The training process of CGAN

In Fig. 2, the server-designated client helps with CGAN training. To mitigate the influence of data heterogeneity on model performance, local model knowledge is applied to train generators[16,22]. However, it is not advisable to train a generator with local model knowledge in the early training process. When a local model converges in the late training process, the prediction performance of this local model becomes more reliable. On the other side, a poorly performing local model can undermine the training of a generator and potentially reduce the overall efficiency of FL. Thus, the performance of a local model can affect the performance of a generator because a generator is trained based on a local model.

The training of a generator can utilize the characteristics of transfer learning, applying knowledge from one task to a related one. leveraging common features from a pre-trained generator model significantly improves the performance of a generator, reduces reliance on large labeled data, expedites training, and enhances generalization on new data.

In the data heterogeneous scenario, each client just considers one domain. So, the idea of transfer learning is applied to train a generator. Before a FL process, each domain uses a CGAN for collaborative training. Due to the non-IID nature of client skew data, the CGAN model trains all domains in an iterative process, which is different to the traditional transfer learning models that train all domains simultaneously. During the training process using the CGAN model, the CGAN model can generate synthetic (image) data that reflects only the final client (domain). Considering that one round of training of unevenly distributed client data can bring data bias and cause a generator to generate the data that are beneficial to the last client, we conduct multiple rounds of training and shuffle the training orders of clients in each round to reduce data fitting bias and better adapt to each client's data.

As shown in Fig. 2, the server randomly generates a sequence including all clients ($c_1, c_2, \cdots, c_N$, $i = 1, 2, \cdots, N$) before each round. Then, the server initializes the CGAN model θ_{CGAN}. After that, the server sends the current CGAN model θ_{CGAN} to the client c_i. The client c_i uses its own data D_{c_i} to train the CGAN model and update the trained model $\text{train}(D_{c_i}, \theta_{\text{CGAN}})$ to the updated model $\acute{\theta}_{\text{CGAN}}$, i.e., $\acute{\theta}_{\text{CGAN}} \leftarrow \text{train}(D_{c_i}, \theta_{\text{CGAN}})$. The trained client c_i sends the updated CGAN model $\acute{\theta}_{\text{CGAN}}$ back to the server. Meanwhile, the server specifies a subsequent client to train the generator $\acute{\theta}_{\text{CGAN}}$ in the client queue order. The above process continues sequentially until all clients complete their respective training. The server then shuffles the entire sequence for the next round of training. Algorithm 1 shows the corresponding detailed steps.

Although the training processing needs longer training time and increased communication overhead, more training rounds typically enhance the

performance of the CGAN model and the quality of synthetic data. Therefore, the proposed framework should keep a balance between training time and (image) data quality.

Algorithm 1: The collaborative training process of CGAN

Input: The number of communication rounds T, the number of clients N, the number of local epochs E, and the clients' local datasets $D_1, \cdots, D_k, \cdots D_N$, C_t is the collection of clients and k is the client in it.

Output: The generator model θ_g and the discriminator model θ_d.

Run on the server:

Initialize θ_g and θ_d;

for $t = 0 : 1 : T-1$ do

$C_t \leftarrow$ (Shuffled list of N clients);

for $k \in C_t$ do

Return the generator trained by the k-th client to the server i.e., θ_g, $\theta_d \leftarrow$ ClientUpdateGenerator (k, θ_g, θ_d);

return θ_g and θ_d;

Update the generator trained by the k-th client, i.e., ClientUpdateGenerator (k, θ_g, θ_d);

for E local epochs do

for Batch $b = \{x, y\} \in D_k$ do

Train the generator θ_g with local data, i.e., $\theta_g, \theta_d \leftarrow$ CGANTraining (b);

return θ_g and θ_d

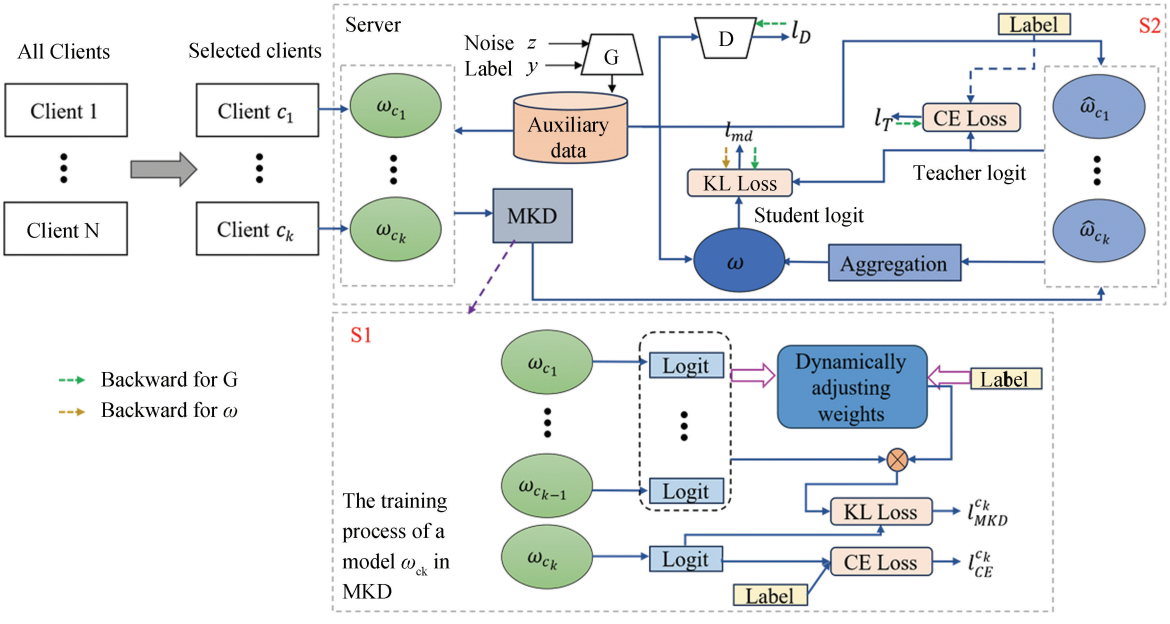

Fig. 3 The FL process of the FedMkd-G framework

In Fig. 3, Block S_2 shows that the generalization performance of the global model is improved using the techniques of mutual KD and generator fine-tuning. Here, MKD represents the mutual KD modules. In addition, Block S_1 introduces the distillation process of a local model in MKD.

1.4 Mutual Learning of Local Models

The FL stage starts after the training of the CGAN model. During the learning process, a generator will stay on the server and not be sent to users to ensure user privacy. In heterogeneous data scenarios, the significantly different local models will improve the suboptimal performance of the aggregated model. Regarding this, all local models will engage in mutual learning before model aggregation, and this engagement can enhance their generalization ability. In Fig. 3, the server uses the generator G to generate the following pseudo data.

$$\tilde{x} = G(z, y; \theta_g) \tag{2}$$

here, θ_g is the parameter of G, $z \sim N(0,1)$ is the standard Gaussian noise, and y is the class label of $\tilde{x}$. In the classical mutual learning process, multiple local models exchange information during training, and the knowledge weights among local models are equal and their contributions are the same. However, the prediction performance of local models for each class will vary with the input of non-IID data. Averaging multiple local models to make predictions integrates knowledge from different clients but can mislead local model training process. This issue is especially serious

when some local models perform poorly on certain classes. Considering the performance or knowledge differences among local models, a weight allocation strategy is applied to balance the contributions of different local models. The distillation weights are calculated according to the samples generated (or predicted) by local models[23].

$$w_{MKD}^{k} = \frac{1}{K-2}(1 - \|L_{CE}^{k}\|^2 / \sum_{j} \|L_{CE}^{j}\|^2) \quad (3)$$

where, parameter K is the distillation weight. w_{MKD}^{k} [9] represents the total local models that are uploaded to the server, and the K−1 local models act as teacher models and guide the training of the remained student model. When each local model carries out mutual KD, the K−1 teacher models are adopted to guide the training of the remained local model. The distillation weight of K − 1 teacher models is represented as Eq. (3). Here, L_{CE}^{k} represents the cross entropy of the k-th client.

$$L_{CE}^{k} = \mathrm{CE}(\sigma(C(\tilde{x};\omega_k)), y) \quad (4)$$

where, CE(cross entropy) denotes the cross-entropy loss, C is a classifier, ω_k is the k-th local model, and σ is the softmax function that can output the predicted score of $\tilde{x}$. In addition, a model with a smaller L_{CE}^{k} will have a larger w_{MKD}^{k}, with a greater contribution to the remained student model.

To efficiently aggregate the predicted samples of multiple local models, different distillation weights are assigned to reflect the predicted samples' confidence after the calculation of the cross-entropy loss between the predicted and labeled samples of local models. Through inversely weighting models based on their cross-entropy loss, FedMkd-G reduces the influence of poorly performing models. The module dynamically adjusting weights (DAW) in Fig. 3 is used for the implementation of Eq. (3).

Next, the workflow of the module DAW in mutual KD is introduced. Block S_1, where the DAW module resides, illustrates the training process of local models for mutual KD. Around the module DAW, the pseudo-data are input into K local models. Then, K−1 local models are used to calculate the KD weights through Eqs. (3, 4), and the obtained KD weights are used to calculate the total teacher models' loss which is taken as the mutual distillation loss of the remained student model. Based on the classical KD, a distillation weight is newly introduced to quantify client contribution, enabling more accurate guidance.

$$L_{\mathrm{MKD}}^{k} = \sum_{l=1, l\neq k}^{K} w_{\mathrm{MKD}}^{l} \, \mathrm{D}_{KL}(\sigma(C(\tilde{x};\omega_l)) \,\|\, \sigma(C(\tilde{x};\omega_k))) \quad (5)$$

here, the parameter $\mathrm{D_{KL}}$ stands for the mutual distillation loss, and L_{MKD}^{k} represents the KL (Kullback-leibler) divergence of KD. The teacher model that is predicted to be closer to the labeled sample will be assigned a greater weight (e.g. a greater w_{MKD}^{k}) because it is confident enough to make a correct instruction. Therefore, the total loss of a local model (e.g. L^{k}) is the sum of this local model's cross-entropy loss and mutual distillation loss.

$$L^{k} = L_{CE}^{k} + L_{\mathrm{MKD}}^{k} \quad (6)$$

The importance of simple samples in guiding model training decreases because of the improvement of model performance during training. To ensure that the generator G provides valuable samples for model training, we use the knowledge among local and global models to fine-tune generators. The local models then use distillation knowledge to assist in the training of the global model. In Fig. 3, the global model is the student model, while the local models act as the role of teacher models. For efficient utilization of the teacher model's knowledge, the distributed valuable samples are explicitly identified and transferred to the global model. Assuming that the teacher model can predict correctly, it is the main goal to differentiate the prediction results of the student and teacher models. The related loss function about prediction is defined as follows in Eq. (7).

$$L_G = \alpha_{\mathrm{g}} L_D + (1 - \alpha_{\mathrm{g}})(L_T - \sum_{\mathrm{k}\in S_{\mathrm{t}}} L_{\mathrm{md}}^{k}) \quad (7)$$

here, L_D is the loss of the original discriminator, L_T is the predicted loss of local models on generated samples, and α_{g} is a hyper-parameter. After fine-tuning a generator, the quality of the newly generated samples is ensured, and the differences between the local models and the global model are observed at the same time. The samples that produce identical prediction results for the local models and the global model do not have training significance for the global model.

$$L_T = \frac{1}{K}\sum_{\mathrm{k}\in S_{\mathrm{t}}} L_{CE}^{k} \quad (8)$$

The parameter L_{md}^{k} is the model difference (md) between the global model ω and the local model ω_{k}, which is measured by KL divergence.

$$L_{\mathrm{md}}^{k} = \mathrm{D}_{KL}(\sigma(\mathrm{D}(\tilde{x};\omega)) \,\|\, \sigma(\mathrm{D}(\tilde{x};\omega_k))) \quad (9)$$

Through Eq.(7), the fine-tuned generator can

produce more valuable samples for the training of the global model by minimizing the following loss.

$$L = \frac{1}{|S_t|}\sum_{k\in S_t} L_{md}^{k} \tag{10}$$

After the training using the knowledge of local models and the samples generated by the fine-tuning generator, the global model converges faster. Algorithm 2 shows the detailed FL process of FedMkd-G.

Algorithm 2: FL process of FedMkd-G

Input: The number of communication rounds U, the generator model θ_g, the discriminator model θ_d, the client sampling ratio C, the local learning rate η, the global learning rate η_g, the external iterations for the updating of a generator and the training of the global model I, the internal iterations for the training of the global model I_g, the iterations for the mutual learning of local models I_m, and the generator learning rate β and α_g. S_t is the collection of clients participating in training, and k is the client.

Output: The final model ω^T.

Run on the server:

Initialize ω;

for $t = 0, \cdots, U-1$ do

Randomly select clients to participate in training, i.e., $S_t \leftarrow$ (randomly selected $C \times N$ clients from $\{c_1, c_2, \cdots, c_N\}$);

for $k \in S_t$ do (in parallel)

Perform the local update operation for the k-th client, i.e., $\omega_k \leftarrow$ ClientUpdate(ω, k);

Perform the local update operation FedAvg, i.e., ▷FedAvg

Perform the mutual learning operation on local models, i.e., $\{\omega_k\}_{k\in S_t} \leftarrow$ ClientMutualLearning (S_t, θ_g);

Aggregate local models into the global model, i.e., $\omega \leftarrow \sum_{k\in S_t} \frac{1}{|D_{S_t}|}|D_k|\omega_k$;

for $i = 1:1:I$ do

Sample labels and noises, i.e., $(Z, Y) \leftarrow$ (a batch of samples and noises z $N(0, 1)$ and y $p(y)$);

Use noises and labels to generate data, i.e., $\tilde{x} = G(Z, Y, \theta_g)$;

Calculate the loss of the generator θ_g, i.e., $L_G = \alpha L_D + (1-\alpha)(L_T - \sum_{k\in S_t} Lk_{md})$;

Update the generator θ_g, i.e., $\theta_g \leftarrow \theta_g - \beta \nabla_{\theta_g} L_G(\theta_g)$;

for $j = 1:1:I_d$ do

Calculate the loss of the global model, i.e., $L = \frac{1}{|S_t|}\sum_{k\in S_t} Lk_{md}$;

Update the global model, i.e., $\omega \leftarrow \omega - \eta_g \nabla_\omega L(\omega)$;

return ω^T;

Perform client mutual learning operation ClientMutualLearning (S_t, θ_g);

for $i = 1:1:I_m$ do

Sample labels and noises, i.e., $(Z, Y) \leftarrow$ (a batch of samples and noises z $N(0, 1)$ and y $p(y)$);

Use noises and labels to generate data, i.e., $\tilde{x} = G(Z, Y, \theta_g)$;

for $k \in S_t$ do

Calculate the cross entropy loss, i.e., $L_{CE}^{k} = CE(\sigma(C(\tilde{x};\omega_k)), y)$;

Calculate the mutual KD loss of the local model ω_k, i.e. $L_{MKD}^{k} = \sum_{l=1, l\neq k}^{K} w^{l}{}_{MKD}\, D_{KL}(\sigma(C(\tilde{x};\omega_l)) \| \sigma(C(\tilde{x};\omega_k)))$;

Calculate the total loss of the local model w_k, i.e., $L^k = L_{CE}^{k} + L_{MKD}^{k}$;

Update the local model w_k, i.e., $\omega_k \leftarrow \omega_k - \eta \nabla L(\omega_k)$;

return $\{\omega_k\}_{k\in S_t}$;

Next, a training case with N clients is given to illustrate the workflow of FedMkd-G. Here, a CGAN model is trained by all clients collaboratively. Initially, the server sets the model parameters for a generator with random values. Then, the server runs T rounds according to Algorithm 1. At the t-th round, the server shuffles a client ID list and sequentially distributes a generator to clients according to the shuffled list. Then, all clients modify their generators based on their local datasets and upload the revised generators to the server. Once the generator training is completed, the federated training process shown in Algorithm 2 will begin. In this stage, the server randomly initializes the global model parameters ω and runs T communication rounds according to Algorithm 2. At the t-th round, $C * N$ clients are sampled by the server to form the sampled client set S_t. The server then sends the related model parameters (e.g. ωt) to the clients in S_t. Using the received parameters, each client in S_t performs local updates and uploads them back to the server. When the server receives all

updated parameters from the clients in S_t, two main tasks are executed. Firstly, the server performs improved mutual KD for each client in S_t. Specifically, the generator θ_g generates auxiliary datasets, and each client calculates these datasets' distillation weights. Then, these distillation weights are uploaded to each client. When each client is updated, the server updates the generator θ_g by minimizing the loss function shown in Eq. (7). After that, the server refines the global model by minimizing the gap between the logits of the global and local models.

2 Results and Discussion

2.1 Experimental Settings

2.1.1 Datasets

The related experiments are conducted on three image datasets, which are MNIST[24], CIFAR10[25], and CIFAR100[25]. The MNIST dataset[22] includes 60,000 training samples and 10,000 test samples, and each sample is a grayscale image of size 28×28. This dataset covers 10 classes (labeled as 0, 1, 2, ⋯, 9) of handwritten numeric images, and all images are evenly distributed in each class. The CIFAR10 dataset[25] has 50,000 training samples and 10,000 test samples. The RGB format of each image is 32×32, and all images evenly distribute across 10 classes. The dataset CIFAR100[25] contains 50,000 RGB training images and 10,000 RGB test images, and all images evenly distribute in 100 classes. Relatively, CIFAR100 is more complex than MNIST and CIFAR10 and is used to measure the stability of FedMkd-G in difficult tasks. The Dirichlet distribution Dir(α) with the same scale is employed to model the non-independent and identically distributed data between clients[11,26,27]. Specifically, we sample $p_k \sim \mathrm{Dir}(\alpha)$ and set the ratio $p_{k,i}$ from the k-th client to the i-th client. Here, α is the concentration parameter that controls the uniformity between clients. When $\alpha \to 0$, the client data shows extreme skewness (such as a client containing only 1-2 types of samples). When $\alpha = 1$, the IID distribution emerges. Fig. 4 shows the statistical heterogeneity across clients on CIFAR-10 with different values of the concentration parameter α. In this figure, the size of the scattered points represents the number of training samples for each labeled client.

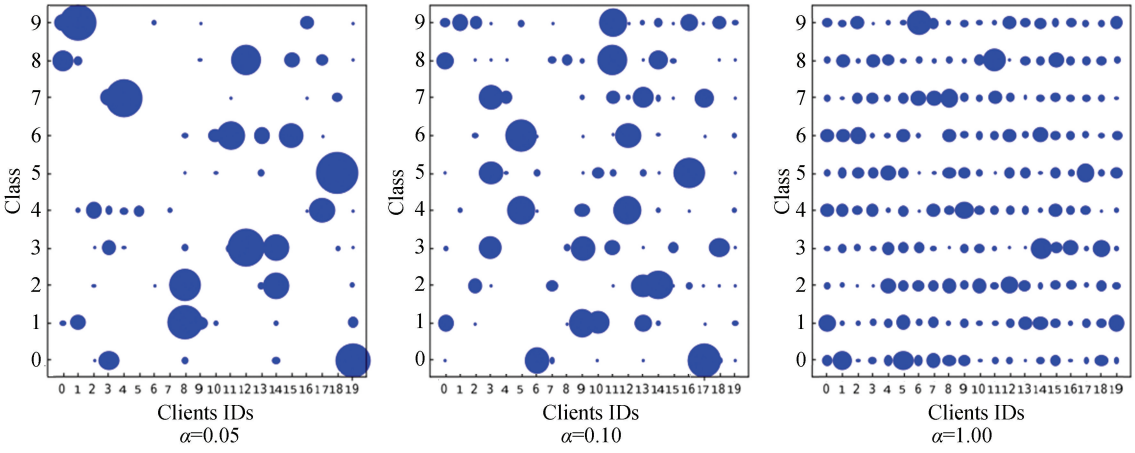

Fig. 4 Visualized statistical heterogeneity among the clients on the CIFAR-10 dataset with different values of α

2.1.2 Models

A simple network architecture is employed for MNIST, which comprises two 5×5 convolutional layers, a fully connected layer with ReLU activation (120 units), and a final softmax output layer. The first convolutional layer has 6 channels, and the second has 16 channels where each layer is followed by ReLU activation and 2×2 max pooling. On the other side, a simple 5-layer CNN network architecture is used for CIFAR10 and CIFAR100, which has three 3×3 convolutional layers including a fully connected layer with ReLU activation (32 units) and a final softmax output layer. The first, second, and third layers have 16, 32, and 64 channels respectively, and each layer is followed by ReLU activation and 2×2 max pooling. Inspired by the traditional architecture[28], we establish the generated network architecture for FedMkd-G.

2.1.3 Configurations

The complete learning process includes 300

global communication rounds, a total of 20 local models, and an active user ratio r (r = 50%). On the datasets MNIST, CIFAR10, and CIFAR100, the corresponding generator has 30, 120, and 150 training epochs respectively. The local update steps are fixed at 20 (E = 20) with 32 small batches per step (B = 32). Up to 50% of the training datasets are allocated to user models, and the entire test datasets are used for performance evaluation.

Adam is adopted as the optimizer for its adaptive learning rate mechanism, which mitigates gradient variances for non-IID data. The learning rates of a classifier and a generator are initialized to 0.01 and 0.0002 respectively, and the exponential decay is set as:

$$\eta_t = \eta_0 \times (0.998)^t \tag{11}$$

here, η_0 is the initial learning rate, and t is the communication round. This strategy gradually reduces the learning rate to stabilize the training in non-IID scenarios. The decay factor 0.998 is chosen to balance exploration (early rounds) and exploitation (late rounds). The z dimensions of MNIST and CIFAR10 are set to 128, and the z dimension of CIFAR100 is set to 256. In Algorithm 2, I_m, I, I_g, and I_d are set to 15, 10, 1, and 5 respectively, and α_g is usually set to 0.5. The proposed FedMkd-G and the introduced baseline methods are implemented using PyTorch[29] and the RTX 3060 GPU.

2.1.4 Baseline methods

FedProx, different to FedAvg, regularizes local model training with proximal terms, while FedSKD is a FL method designed for selective KD in bandwidth-constrained wireless networks. FedAKD is a lightweight FL method designed for enhancing KD and human activity recognition. FedGen is a federated distillation method with flexible parameter sharing. In Fig. 5, the results of global test accuracy indicate the global model's performance on the test datasets. Here, Dir (α = 0.1) is the default selection because it balances heterogeneity and convergence efficiency.

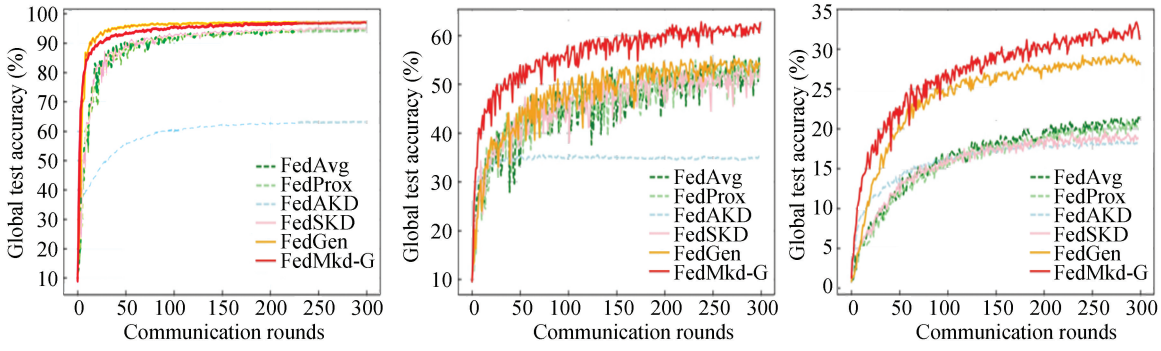

Fig. 5 The learning curves on the benchmark datasets with default settings

2.2 Performance Analysis

2.2.1 Accuracy

The learning curves of the global model are shown in Fig. 5, and Table 1 shows the detailed performances across various data settings, a smaller α indicates higher heterogeneity, and T denotes the local training steps. On MNIST with α = 0.05 (extreme heterogeneity), FedMkd-G achieves the result of 95.16% in accuracy, which outperforms FedAvg by 6.2%. Fig. 5 shows that FedMkd-G converges after 22 rounds where α = 0.1, which requires fewer rounds than FedAvg. This efficiency improvement stems from the α-controlled data heterogeneity. Specifically, a smaller α amplifies the need for GAN-generated data to bridge distribution gaps, while mutual KD adapts to a varying α via dynamic weight allocation (Eq. 3). On CIFAR10, FedMkd-G outperforms all baseline methods in terms of final model test accuracy and convergence speed (measured by communication rounds) and demonstrates its effectiveness for non-IID data and shallow neural networks. Similar conclusions are drawn according to the results on CIFAR100. These conclusions certify the robustness of FedMkd-G for various datasets and model architectures. In contrast, FedAvg struggles to converge and find a global optimum in non-IID environments. FedProx also faces slower client updates in keeping closely updated directions. FedAKD faces notable performance degradation. Although FedSKD outperforms FedAvg and FedProx on MNIST, its accuracies on CIFAR10 and CIFAR100 are slightly lower. FedMkd-G trains the local and global models on the server side using auxiliary datasets, which helps the global model converge faster at the

beginning and utilizes a pre-trained generator effectively. This operation benefits all clients, helps the global model approach the globally optimal state, and makes it more possible for a local optimal state to evolve into the global optimal state. As a result, all clients naturally have similar update directions of encouraging convergence and can create a virtuous cycle.

Table 1 Performance overview with different data settings

Dataset	Setting	Top-1 test accuracy					
		FedAvg	FedProx	FedAKD	FedSKD	FedGen	FedMkd-G
MNIST $T=20$	$\alpha=0.05$	88.96±3.26	88.99±2.85	60.45±0.60	90.35±2.35	94.69±0.23	**95.16±0.17**
	$\alpha=0.10$	94.59±0.17	94.53±0.25	63.21±0.23	95.98±0.30	**97.21±0.04**	97.15±0.03
	$\alpha=1.00$	97.24±0.09	97.12±0.09	83.71±0.48	97.46±0.21	**98.27±0.04**	98.24±0.18
CIFAR10 $T=20$	$\alpha=0.05$	49.73±0.76	49.15±0.83	45.11±0.21	49.48±1.23	51.54±1.41	**60.69±0.45**
	$\alpha=0.10$	54.91±0.73	54.76±0.54	35.64±0.22	53.75±0.29	55.98±0.49	**62.71±0.09**
	$\alpha=1.00$	63.06±0.67	62.86±0.53	35.43±0.61	62.73±0.50	65.38±0.12	**65.49±0.25**
CIFAR100 $T=20$	$\alpha=0.05$	20.70±0.32	20.09±0.75	22.38±0.04	18.37±0.70	27.09±0.21	**32.70±0.35**
	$\alpha=0.10$	21.81±0.87	21.36±0.69	18.80±0.51	20.68±1.08	29.09±0.59	**33.20±0.15**
	$\alpha=1.00$	24.60±0.81	23.76±0.68	12.75±0.29	24.04±0.95	32.56±0.67	**33.98±0.29**

Additionally, the test accuracy of several FL methods is illustrated, along with the communication rounds necessary to reach the target test accuracy (90% for MNIST, 50% for CIFAR10, and 19% for CIFAR100). In Table 2, FedMkd-G achieves the second-best results on MNIST and delivers the best results on CIFAR10 and CIFAR100, which outperforms all baseline methods with a notably faster convergence rate. The best and the second-best values are highlighted and $\alpha=0.1$. These findings demonstrate FedMkd-G's capability in speeding up global model convergence.

Table 2 Accuracy evaluation of different FL methods using the communication rounds required by the target testing accuracy on MNIST, CIFAR10, and CIFAR100

	MNIST T (90%)	CIFAR10 T (50%)	CIFAR100 T (19%)
FedAvg	49	114	155
FedProx	60	138	199
FedAKD	–	–	–
FedSKD	48	135	201
FedGen	**13**	89	42
FedMkd-G	22	**33**	**28**

2.2.2 Impacts of data heterogeneity levels

To evaluate the FedMkd-G's robustness under different levels of data heterogeneity, the Dir(α) strategy based on the Dirichlet distribution is used on CIFAR10. Table 3 demonstrates the robust performance of FedMkd-G across α values on CIFAR10. When $\alpha=0.2$, FedMkd-G reaches 63.73% accuracy and outperforms FedGen by 3.8% and FedAvg by 5.6%. These results validate that GAN-generated data can mitigate client drift from label skewness. When α rises from 0.2 to 1.0, the accuracy growth for FedMkd-G is 1.76%, which is significantly less than the one 5.45% for FedGen. This result indicates that FedMkd-G is less reliant on data uniformity and makes FedMkd-G suitable for real-world non-IID cases. FedMkd-G consistently achieves optimal accuracy values for all settings, and these superior results confirm the effectiveness of FedMkd-G in diverse data heterogeneous scenarios. Particularly, FedMkd-G gets higher accuracy improvements in the extreme data heterogeneity scenario where $\alpha=0.2$. Moreover, the accuracy of each method increases when the degree of data heterogeneity decreases or the parameter α increases. Generally, the above experimental analysis demonstrates the robustness and effectiveness of FedMkd-G at different heterogeneity levels.

2.2.3 Impacts of client participation rates

To evaluate the scalability of FedMkd-G, each communication round on CIFAR10 involves a different number of participating clients. The CIFAR10 training dataset is assigned to 20 clients, and 5, 10, 15, or 20 clients are randomly chosen to participate in each round. As shown in Fig. 6, FedMkd-G consistently outperforms other methods even in scenarios with fewer participants per round. This superior performance demonstrates the robustness of FedMkd-G in the cases with different levels of client participation.

2.2.4 Impacts of local update epochs

The aggregation of local models using different

frequencies can impact learning performance because lower-frequency communication can exacerbate drift during the local training stage. The related experiments are conducted to examine the effect of local epochs on the final global model performance. As depicted in Fig. 6, a smaller E can increase the communication burden, while a larger E can lead to slower convergence rates. When $E=1$, the test accuracies of all methods slightly decrease because each client frequently communicates with others to minimize the negative effect of "client drift". Given more local epochs, each local client further updates drift and widens the gap between the local and global models. When $E=50$, the performances of FedAvg and FedProx degrade. However, FedMkd-G facilitates mutual learning among users, mitigates the potential distribution differences among users, and concurrently integrates the knowledge of local models into the global model. FedMkd-G thus outperforms the other methods, which demonstrates its ability to withstand the significant drift resulting from more locally updated epochs.

Table 3 The top-1 test accuracy with different levels of data heterogeneity on CIFAR10

Method	$\alpha=0.2$	$\alpha=0.4$	$\alpha=0.6$	$\alpha=0.8$	$\alpha=1.0$
FedAvg	58.16	61.25	61.60	63.38	63.06
FedProx	58.69	60.63	61.20	63.35	62.86
FedAKD	32.29	30.93	32.53	34.62	35.41
FedSKD	57.36	60.23	60.82	62.46	63.72
FedGen	59.93	62.79	63.73	65.12	65.38
FedMkd-G	**63.73**	**64.71**	**64.60**	**65.23**	**65.49**

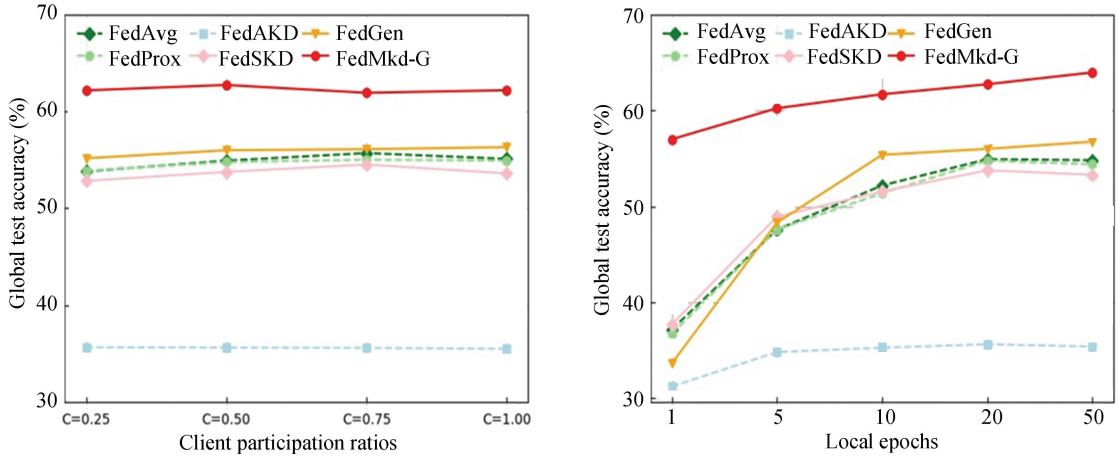

Fig. 6 Visualized performance of the global models on CIFAR10 with different client participation ratios and different numbers of local epochs

2.2.5 *Ablation experimental results*

The ablation experiments are conducted to examine the key components in FedMkd-G. Table 4 shows the accuracies of FedMkd-G after removing certain modules. On CIFAR10, the training process here includes over 300 communication rounds with $\alpha=0.1$. In Table 4, dak, mkd, and ftg are the abbreviations of dynamic adjustment of distillation weights, local model mutual learning, and fine-tuning generators respectively, and the related experiments are conducted on CIFAR10, and $\alpha=0.1$. It is evident that the removal of any module results in a corresponding decrease in accuracy. These experimental results highlight the effectiveness of these key components in FedMkd-G.

Table 4 Impacts of the key components in FedMkd-G

	Method	Accuracy (%)
Baseline	FedMkd-G	62.71
Module	-dak	62.30
	-mkd	59.05
	-ftg	61.71

The experimental results demonstrate the advantages of FedMkd-G, and these advantages certify that FedMkd-G is a promising method for FL. Generally, FedMkd-G can effectively handle data

heterogeneity and shows superior generalization performance in scenarios with variant data distributions compared with other conventional methods. The robustness of FedMkd-G is suitable for different datasets and model architectures. So, FedMkd-Ghas a potential generalization to various real-world applications. Additionally, FedMkd-G can not only accelerate the global model's convergence but also require fewer communication rounds to reach the desired accuracy compared to the introduced baseline methods.

On the other side, FedMkd-G has itslimitations. FedMkd-G will incur increased computational overhead due to introduced generators and additional training stages. Especially, FedMkd-G needs more computational overhead in scenarios with a large number of clients. Comparatively, FedMkd-G also spends more training time. Therefore, future work should prioritize the development of scalable training strategies that improve performance while lower training time and computational burden. In addition, a more efficient generator design can reduce computational overhead and make FedMkd-G more feasible for real-world applications.

2.2.6 Computational overhead analysis

In the context of FL, it is crucial for practical deployment to understand the computational overhead and training time of different algorithms. Table 5 presents a computational overhead comparative analysis of FedAvg, FedProx, and FedMkd-G under the CIFAR10 dataset with $\alpha = 0.1$. On the CIFAR10 dataset where $\alpha = 0.1$, FedAvg and FedProx have no extra computational overhead, while FedMkd-G needs additional costs, i.e., 24.5×10^9 FLOPs for CGAN pre-training and 18.2×10^6 FLOPs per-round distillation. Regarding training time, FedAvg requires 0.5 h and 114 rounds to converge, while FedProx requires 0.53 h and 138 rounds to converge. On the other hand, FedMkd-G requires a relatively longer training time (1.25 h). However, it converges after only 33 rounds. Despite the longer total training time for FedMkd-G, its faster convergence rate is a significant advantage. In practical scenarios, e.g., the scenario of dealing with large-scale datasets or a large number of clients, the reduced number of rounds can potentially offset the additional time cost per round.

Table 5 Computational overhead (The related experiments are conducted on CIFAR10, and $\alpha = 0.1$)

Comparison dimension	FedAvg	FedProx	FedMkd-G
Computational overhead	No additional	No additional	CGAN pre-training: 24.5×10^9 FLOPs Per-round distillation: 18.2×10^6 FLOPs
Training time	0.5 h (114 rounds)	0.53 h (138 rounds)	1.25 h (converges in 33 rounds)

In conclusion, the faster convergence rate of FedMkd-G provides potential benefits for practical deployment although it does have additional computational overhead compared to FedAvg and FedProx. Considering the trade-off between computational cost and convergence efficiency and optimized training strategies, FedMkd-G can be a better option in non-IID FL scenarios. Especially, FedMkd-G is suitable for the scenario where high model convergence and generalization for heterogeneous data are required.

3 Conclusions

Here, a new FL method FedMkd-G based on KD is proposed, which is designed to address user heterogeneity without relying on external data sources. Specifically, FedMkd-G dynamically adjusts distillation weights among local models during the mutual learning process and enhances the exploration level of local knowledge. The refined knowledge gained from exploration is then transferred to local models for additional training, and the generalization performance of the local models and the overall performance of the combined global model is then enhanced.

Furthermore, FedMkd-G fine-tunes a generator to identify valuable samples for the global model training and can maximize the utility of local model knowledge. In general, FedMkd-G can significantly increase the efficiency of knowledge transfer and model refinement. After extensive experiments on three benchmark datasets, the results confirm the effectiveness of FedMkd-G in enhancing model performance and demonstrate the potential of FedMkd-G for future research and application development.

In summary, FedMkd-G can effectively address data heterogeneity in FL through mutual KD and GAN-generated auxiliary data, which demonstrates significant advantages in the following practical

scenarios.

(1) Medical FL. In the scenarios with extreme data heterogeneity (e. g., disease distribution disparities across hospitals, $\alpha \approx 0.05$), FedMkd-G achieves 60.69% accuracy on CIFAR10 (Table 1), and this accuracy is 21.96% higher than the one for FedAvg. The GAN-generated synthetic samples supplement rare disease data, while mutual KD aligns diagnostic knowledge across institutions. These advantages reduce communication rounds by 71% (from 114 rounds to 33 rounds), which are crucial for medical data transmission.

(2) Cross-regional recommendation systems. For dynamic user behavior data (e.g., regional consumption preferences, $\alpha = 0.1$), FedMkd-G outperforms FedDF by 13.57% in accuracy on CIFAR100 (shown in Table 1). The dynamic distillation weights prioritize high-confidence regional models (e. g., northern winter consumer patterns), while GAN-generated cross-regional features provide FL training data.

References

[1] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Cambridge: MIT Press, 2007,54:1273-1282.

[2] Khaled A, Mishchenko K, Richtárik P. Tighter theory for local SGD on identical and heterogeneous data. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. Cambridge: MIT Press, 2020. 108: 4519-4529.

[3] Li X, Huang K, Yang W, et al. On the convergence of fedavg on Non-IID data. 2020 International Conference on Learning Representations. arXiv: 1907. 02189. 2020. DOI: 10.48550/arXiv.1907.02189.

[4] Li T, Sahu A K, Talwalkar A, et al. Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 2020. 37(3):50-60. DOI: 10.1109/MSP.2020.2975749.

[5] Zhao Y, Li M, Lai L, et al. Federated learning with Non-IID data. arXiv:1806.00582. 2018. DOI: 10.48550/arXiv.1806.00582.

[6] Karimireddy S P, Kale S, Mohri M, et al. Scaffold: stochastic controlled averaging for federated learning. Proceedings of the 37th International Conference on Machine Learning. New York: ACM Press, 2020, 476: 5132-5143.

[7] Dinh C T, Tran N H, Nguyen T D. Personalized federated learning with moreau envelopes. Proceedings of the 34th International Conference on Neural Information Processing Systems. New York: ACM Press, 2020, 1796: 21394-21405.

[8] Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems. Indio: MLSys, 2020.

[9] Seo H, Park J, Oh S, et al. Federated knowledge distillation.arXiv:2011.02367. 2020. DOI: 10.48550/arXiv.2011.02367.

[10] Chen H Y, Chao W L. FedBE: making bayesian model ensemble applicable to federated learning. International Conference on Learning Representations. Appleton: ICLR, 2021.

[11] Yurochkin M, Agarwal M, Ghosh S, et al. Bayesian nonparametric federated learning of neural networks. International Conference on Machine Learning. Cambridge: MIT Press, 2019,97:7252-7261.

[12] Li D, Wang J. FedMD: Heterogeneous federated learning via model distillation. arXiv:1910.03581. 2019. DOI: 10.48550/arXiv.1910.03581.

[13] Lin T, Kong L, Stich S U, et al. Ensemble distillation for robust model fusion in federated learning. Proceedings of the 34th International Conference on Neural Information Processing Systems. New York: ACM Press, 2020, 198: 2351-2363. DOI: 10.5555/3495724.3495922.

[14] Gad G, Fadlullah Z M, Fouda M M, et al. Federated learning with selective knowledge distillation over bandwidth-constrained wireless networks. Proceedings of the IEEE International Conference on Communications. Piscataway: IEEE, 2024, 3476-3481. DOI: 10.1109/ICC51166.2024.10622906.

[15] Gad G. Light-weight Federated Learning with Augmented Knowledge Distillation for Human Activity Recognition. Thunder Bay: Lakehead University, 2023.

[16] Zhu Z, Hong J, Zhou J. Data-free knowledge distillation for heterogeneous federated learning. Proceedings of the 38th International Conference on Machine Learning. Cambridge: MIT Press, 2021, 139: 12878-12889.

[17] Zhang L, Shen L, Ding L, et al. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2022, 10164-10173. DOI: 10.1109/CVPR52688.2022.00993.

[18] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv: 1503. 02531. 2015. DOI: 10.48550/arXiv.1503.02531.

[19] Zhang Y, Xiang T, Hospedales T M, et al. Deep mutual learning. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018, 4320-4328. DOI: 10.1109/CVPR.2018.00454.

[20] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets. Proceedings of the 28th International Conference on Neural Information Processing

Systems. New York: ACM Press, 2014, 2: 2672-2680. DOI: 10.5555/2969033.2969125.

[21] Zhuang W, Wen Y, Zhang X, et al. Performance optimization of federated person reidentification via benchmark analysis. Proceedings of the 28th ACM International Conference on Multimedia. New York: ACM Press, 2020, 955-963. DOI: 10.1145/3394171.3413814.

[22] Khoussainov R, Heβ A, Kushmerick N. Ensembles of biased classifiers. Proceedings of the 22nd International Conference on Machine Learning. New York: ACM Press, 2005, 425-432. DOI: 10.1145/1102351.1102405.

[23] Zhang H, Chen D, Wang C. Confidence-aware multi-teacher knowledge distillation. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway: IEEE, 2022, 4498-4502. DOI: 10.1109/ICASSP43922.2022.9747534.

[24] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278-2324. DOI: 10.1109/5.726791.

[25] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. Technical Report, 2009.

[26] Acar D A E, Zhao Y, Matas R, et al. Federated learning based on dynamic regularization. International Conference on Learning Representations (ICLR). Appleton: ICLR, 2020.

[27] He C, Li S, So J, et al. FedML: A research library and benchmark for federated machine learning. arXiv preprint arXiv: 2007. 13518. 2020. DOI: 10. 48550/arXiv. 2007. 13518.

[28] Miyato T, Koyama M. cGANs with projection discriminator. arXiv preprint arXiv: 1802. 05637. 2018. DOI: 10.48550/arXiv.1802.05637.

[29] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 2019. 32:8024-8035.