This is the Pre-Published Version.
The following publication D. Xiao, J. Li, X. Mo, W. Wu and J. Cao, "Decentralized Federated Distillation With Protected Pruned Models in Few Global Epochs," in IEEE Transactions on Dependable and Secure Computing, vol. 23, no. 1, pp. 718-734, Jan.-Feb. 2026 is available at https://doi.org/10.1109/TDSC.2025.3610087.

Decentralized Federated Distillation with Protected Pruned Models in Few Global Epochs

Danyang Xiao, Jialun Li, Xuan Mo, Weigang Wu, *Member, IEEE,* Jiannong Cao, *Fellow, IEEE*

***Abstract*—Decentralized Federated Distillation (DFD) has emerged as a significant research direction since it not only supports heterogeneous model training but also naturally avoids privacy risks and communication bottlenecks stemming from the central server. DFD shows great potential in various training scenarios, especially in cross-silo federated environments. Existing DFD algorithms have limitations, including reliance on public datasets for distillation, insufficient privacy protection, and high communication overhead. This paper introduces Ring-Distill, a novel DFD algorithm designed specifically for cross-silo federated environments, effectively addressing the aforementioned limitations. Ring-Distill allows clients to complete federated training within a few global epochs (i.e., communication rounds) without sharing public datasets, e.g., N global epochs for a system involving N clients. To protect privacy and further reduce communication overhead, Ring-Distill contains a privacy-oriented automatic model pruning mechanism (PAMP) which can automatically create the privacy-preserving compressed model called proxy model for each client. These proxy models are employed for client-to-client distillation during each global epoch. Furthermore, Ring-Distill contains a historical model-based distillation (HMD) mechanism that allows local models to transfer knowledge from multiple historical proxy model replicas stored locally. Due to historical proxy model replicas, the HMD mechanism not only improves the distillation performance but also effectively mitigates client dropout issues. Theoretical analysis and comprehensive experiments show that Ring-Distill has significant advantages in terms of accuracy, privacy, and communication cost.**

***Index Terms*—Decentralized Federated Distillation, Federated Learning, Public Dataset-Free Distillation, Model Compression**

I. Introduction

Federated Distillation (FD) [1], [2] (i.e., Distillation-Based Federated Learning) attracts many researchers' attention since it can not only achieve comparable performance to typical Federated Learning (FL) [3], [4], but also support training among heterogeneous models. Through Knowledge Distillation (KD), FD does not require local models participating in FL to have the same size, structure, or even model types.

Decentralized Federated Distillation (DFD) [5], [6], as a variant of FD, has become a notable topic. DFD not only retains the advantages of FD but also naturally avoids privacy risks and communication bottlenecks stemming from the central server [7]. DFD shows significant potential in various applications, particularly in cross-silo federated training, where selecting a central coordinator within a multi-institution collaborative alliance may be challenging.

From the perspective of data exchange, existing DFD variants can be categorized into two paradigms: logits-based DFD and model-based DFD. In logits-based DFD algorithms, such as CMFD [5], participating clients typically facilitate knowledge transfer among their local models by exchanging logits related to public datasets. In scenarios with a limited number of public data samples, logits-based DFD algorithms can effectively reduce the communication data volume. Yet, logits-based DFD algorithms require a public dataset to allow clients to compare each logits one by one with identical batches, which is impractical or faces inherent limitations in some real-world applications. For instance, in sensitive applications such as medical image analysis, acquiring a suitable and high-quality public dataset presents a significant challenge.

Model-based DFDs [6], [8], [9] can perform distillation procedures without relying on a public dataset by exchanging model parameters. In general, model-based DFDs enable clients to align their local model outputs with outputs of received models on private data instead of public data. However, model-based DFDs may encounter certain challenges. Firstly, the communication overhead on the client side is correlated with the size of the trained model and escalates with the growing number of global epochs. Additionally, as these DFDs require exposing models trained on private data to other clients, they inherently pose privacy risks. For instance, honest-but-curious adversaries may employ "white-box" attacks to infer or even reconstruct private data from the received model [10].

To address the issues of existing DFDs [11]–[13], such as dependency on public data, privacy leakage, and high communication overhead, we propose a two-stage, model-based decentralized federated distillation algorithm for cross-silo settings, named Ring Distillation (**Ring-Distill**), which combines the advantages of two DFD paradigms. Different from existing model-based DFDs, the basic idea of Ring-Distill is to enable each client to create its **proxy model** by pruning its converged local model, and to allow clients to transfer knowledge to each other through proxy models instead of local models, without sharing the public dataset. Specifically, Ring-Distill consists of Local Sparse Training Stage and Knowledge Transfer Stage. Each client trains its local model on private data until convergence during the first stage, and then prunes its local model to derive the proxy model used in the second stage. Under Ring-Distill settings,

Danyang Xiao and Jiannong Cao are with the Department of Computing, The Hong Kong Polytechnic University (PolyU), Hong Kong.
E-mail: danyang.xiao@polyu.edu.hk
Jialun Li, Xuan Mo and Weigang Wu are with the School of Computer Science and Engineering, Sun Yat-sen University, and Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, Guangzhou, China.
E-mail: wuweig@mail.sysu.edu.cn

clients can perform knowledge transfer in a public data-free manner like traditional model-based FDs, and all local models remain inaccessible to the public, which helps reduce privacy risks associated with local models.

To further reduce communication overhead, Ring-Distill incorporates an aggregation process[1] based on ring-topology communication, allowing clients to complete collaborative training within just a few global epochs in the second stage. For instance, each client only needs to initiate N communication rounds under Ring-Distill settings involving N clients. However, the proxy model still poses some privacy risks. Therefore, Ring-Distill provides a **P**rivacy-oriented **A**utomatic **M**odel **P**runing (**PAMP**) mechanism. PAMP can automatically prune the local model during sparse training with constraints of both performance and privacy such as inference gain. Furthermore, Ring-Distill introduces a **H**istorical **M**odel-based **D**istillation mechanism (**HMD**) that utilizes previously received proxy models to guide distillation operations. As a result, Ring-Distill not only enhances the distillation performance but also effectively mitigates client dropout issues by leveraging historical proxy model replicas. Our analysis and experimental results show that Ring-Distill achieves the goals of public dataset-free distillation, privacy protection, and efficient communication. **Our contributions are as follows:**

- A two-stage, model-based DFD algorithm called Ring-Distill is proposed, which can achieve the goals of public dataset-free distillation, privacy protection, and efficient communication.
- A privacy-oriented automatic model pruning mechanism called PAMP is designed for Ring-Distill. PAMP can obtain the pruned model that keeps privacy and accuracy.
- Theoretical analysis and comprehensive experiments regarding Ring-Distill are provided. Our evaluation demonstrates the effectiveness of Ring-Distill.

TABLE I
COMPARISON OF EXISTING TYPICAL DFDS

Methods	CMFD	DCCR	Def-KT	DFLStar	FedDCM	Ours
Categories	logits	logits	models	models	models	models
Public Data-Free Distillation	×	×	✓	✓	✓	✓
Privacy Protection Mechanism	×	×	×	×	×	✓
Comm. Optimization	✓	✓	✓	✓	×	✓

II. RELATED WORK

Federated Distillation (FD) [1], [2], [14], a popular variant of Federated Learning [3], [15] has attracted researchers' attention due to its ability to support heterogeneous model training. According to the architectural design, existing FDs can be classified into two major paradigms: centralized federated distillation and decentralized federated distillation.

[1]In general, the aggregation process in FD refers to the distillation process.

A. Centralized Federated Distillation

Knowledge Distillation (KD) enables a moderately performing model to improve its performance by mimicking the outputs of a high-performing model. Inspired by KD, some researchers designed FL frameworks based on KD. FedMD [16], one of the typical FD algorithms, enables clients to upload logits related to public datasets to the server for aggregation, rather than model parameters. FedMD allows the server to calculate the average logits based on the received logits. Average logits are then sent back to the clients. Clients utilize these average logits to align the outputs of their local models. Besides, many variants of FedMD emerged [17]–[19]. For instance, FedDF [20] aggregates the global model by combining FedAvg and KD so as to improve the effectiveness of FD. To reduce communication data volume, researchers design a two-tier knowledge transfer framework called FedKT [21] that can effectively perform private one-shot FL. Recently, to address the catastrophic forgetting issue, researchers proposed a multi-domain FD algorithm called MUFTI [22] that combines knowledge distillation and continual learning. To sum up, centralized FDs mentioned above may face privacy risks and communication bottlenecks stemming from the centralized server. In this paper, we focus on decentralized FD.

B. Decentralized Federated Distillation

Since decentralized FDs can reduce server communication bottlenecks and mitigate server-side attack risks, recently many researchers have paid attention to DFD algorithms. Codistillation [23] is one of the earliest algorithms that employ distillation techniques in decentralized training. Under Codistillation setting, several models are collaboratively trained and then distill knowledge from each other. Inspired by Codistillation, many DFD algorithms are proposed. Existing DFD variants can be categorized into two paradigms: logits-based DFD and model-based DFD.

Logits-based DFD. CMFD [5] is one of the typical logits-based DFD algorithms. Similar to centralized FedMD [16], under CMFD settings, during each communication round, local models of clients transfer knowledge to each other by exchanging logits related to the public dataset. Thanks to the exchanged logits, CMFD can effectively reduce the volume of communication data during training. Building upon CMFD, some researchers proposed DCCR, a variant that incorporates a dynamic communication reduction mechanism to further reduce communication data volumes [24]. However, CMFD and DCCR require a public dataset to allow clients to compare each logit one by one with identical batches, which is impractical in some real-world applications.

Model-based DFD. Def-KT [8], as one of the representative algorithms of model-based DFD, naturally avoids the use of a public dataset since it allows clients to share their local models for mutual knowledge transfer. FedDCM [6] also employs mutual learning techniques for knowledge transfer among clients. Under FedDCM settings, each client maintains two models: the local model and the global model. In particular, the global model is used for distillation operations when sent to other clients. In addition, some studies have proposed DFD

algorithms similar to FedDCM [6], [9]. Both Def-KT and FedDCM face limitations: privacy risks and communication costs. Regarding privacy risks, Def-KT has to expose clients' local models while FedDCM lets the client train the global model on private data. In practice, local models in Def-KT and global models in FedDCM may suffer from while-box attacks. Regarding communication costs, both Def-KT and FedDCM require several communication epochs and send/receive model parameters during each communication epoch, which may cause more communication data volumes. Recent work introduced DFLStar [25], a novel DFD algorithm that minimizes communication overhead by selecting only the most informative neighbors (identified via last-layer model similarity) for model exchange. DFLStar still fails to effectively address privacy concerns.

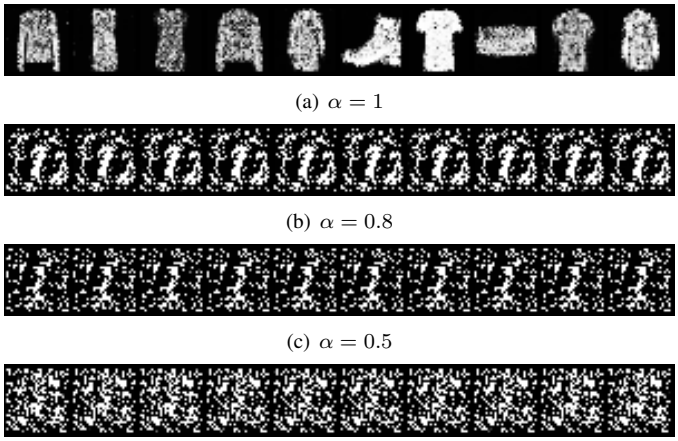
Existing DFD algorithms may have their respective pros and cons. In general, logits-based DFDs have the advantage of low communication data volumes, while model-based DFDs allow clients to transfer knowledge to each other in a public dataset-free manner. However, both, especially model-based DFDs, lack attention to privacy issues during training. In this paper, the proposed Ring-Distill combines the advantages of two DFD paradigms. Table I demonstrates the superiority of our proposed algorithm compared to some typical algorithms.

III. Motivation

The goal of our work is to design a DFD algorithm that reduces **1)** communication overhead and **2)** privacy risks while **3)** enabling knowledge transfer without relying on public data. To meet the above requirement 3, i.e., public data-free distillation, we only focus on model-based DFDs, as these algorithms can perform distillation operations without public data. The core idea of our design is to enable knowledge transfer between clients through "proxy models" rather than their local models. Inspired by the following observations, our work incorporates a privacy-oriented model pruning mechanism to compress clients' local models into proxy models, effectively reducing privacy risks and communication overhead during training.

Observation 1: compressed models can effectively reduce the risk of data reconstruction attacks.

To assess the model's ability to resist data reconstruction attacks, we design experiments that let an attacker steal information by GAN from compressed models with different compression settings. In practice, such an attacker may be an honest-but-curious client trying to capture information from the received compressed model. In our simulation, these compressed models are pruned from converged CNN trained on Fashion-MNIST. Experimental results are shown in Figure 1. In Figure 1, a lower α value indicates a higher compression level. From the results, we believe that, exposing uncompressed models (i.e., $\alpha = 1$) poses significant risks, as attackers may reconstruct private information from these models. For instance, as shown in Figure 1(a), the attacker can reveal the raw data about Fashion-MNIST samples. In contrast, compressed models with different compression rates(α) are strong ability to resist data reconstruction attacks as shown in Figure 1(b-d).



(a) $\alpha = 1$

(b) $\alpha = 0.8$

(c) $\alpha = 0.5$

(d) $\alpha = 0.1$

Fig. 1. Data reconstruction attacks on compressed models with different compression rates (α). A smaller α indicates a higher compression level. Compressed models are pruned from converged CNN trained on Fashion-MNIST.

Observation 2: compressed models demonstrate the potential to resist membership inference attacks. (MIAs).

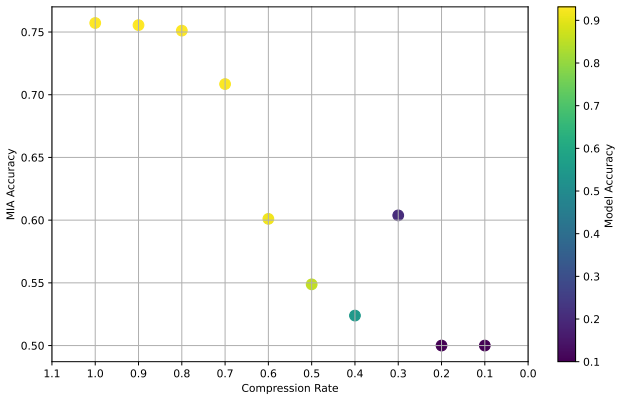


Fig. 2. Membership inference attacks on compressed models with different compression rates. MIA Accuracy refers to the accuracy of the attack model, where higher values indicate greater attack success rates. Different dots' color indicates different compressed model (victim) accuracy. Compressed models are pruned from converged ResNet-18 trained on CIFAR-10.

To assess the model's ability to resist membership inference attacks[2], we design experiments to simulate an attack environment[3] that lets the attacker perform the membership inference attack to infer whether CIFAR-10 samples are trained on the target (compressed/uncompressed) model (ResNet-18). From the results shown in Figure 2, we observe that, except for one outlier, higher compression generally boosts attack resistance (i.e., lower MIA accuracy is better) but diminishes model accuracy. However, we found that some compressed models ($\alpha = 0.5, 0.6$) can improve their resistance to MIAs while maintaining desirable accuracy. From the results, we believe that, compressed models have the potential to resist membership inference attacks. We can explore advanced model

[2]In general, MIA involves training a binary classifier to determine whether private data contains a specific sample.

[3]For generality, the L1 norm-based model pruning API provided by Pytorch is adopted in our evaluation.

pruning techniques to ensure that compressed models maintain both performance (higher accuracy of compressed models) and privacy (lower MIA accuracy of MIA models).

The above observations inspire us to consider: can we leverage the potential privacy protection of compressed models to enhance the effectiveness of DFD algorithms? Existing work [26] starts to explore the above question. In practice, it is challenging for adversaries to carry out data reconstruction attacks. Instead, membership inference attacks (MIAs) pose a more common threat in real-world applications. Thus, we believe that compressed models capable of resisting MIAs have the potential to mitigate privacy risks in real-world scenarios. In this paper, we focus on developing a privacy-oriented model pruning mechanism (i.e., PAMP) and introduce Ring-Distill, which is built upon PAMP.

IV. The Proposed Ring-Distill

We first present the procedure of Ring-Distill, followed by a detailed explanation of its core components: Privacy-Oriented Automatic Model Pruning (PAMP) and Historical Model-Based Distillation (HMD) mechanisms.

A. The Overview of Two-Stage Training Procedure

Algorithm 1 and Figure 3 describe the procedure of Ring-Distill. The entire procedure consists of two stages: 1) Local Sparse Training and 2) Knowledge Transfer.

Local Sparse Training. Each client trains its local model on the private dataset until convergence, and then creates a proxy model from its local model using PAMP. Specifically, after several iterations of sparse training, PAMP prunes the local model to create the proxy model which maintains privacy and accuracy by the constraint of inference gain and sparse training loss. The proxy model is then prepared for transmission to other clients for distillation during the knowledge transfer stage. As a result, all original local models are not exposed to the public during collaborative training, which may avoid privacy risks from local models.

Knowledge Transfer. All participating clients communicate with their neighbors in parallel via the ring-topology-based communication structure. Specifically, during each epoch, each client receives the upstream neighbor's proxy model, and sends its proxy model to the downstream neighbor (Lines 9-10 in Algorithm 1). For each client, after receiving a proxy model, the client applies a designed historical model-based distillation (HMD) mechanism to align its local model's outputs with the proxy model's outputs on its private dataset. The operational details of HMD will be introduced in the following subsection. After distillation, each client sets the received proxy model as the new version of its proxy model, while the old proxy model is stored locally for HMD operations in the next epoch. In subsequent epochs, the client repeats the above operations until its local model has learned knowledge from proxy models of all other clients, that is, when a client has leveraged all proxy models from all other clients, it completes the knowledge transfer stage and can exit the Ring-Distill training procedure (Lines 7-15 in Algorithm 1).

During the entire procedure under RingDistill, without sharing any local model and public dataset, each client can indirectly learn knowledge from local models of other clients through their corresponding proxy models, which can improve the performance of all local models in DFD system. It is important to note that proxy models do not retain sensitive information from the private dataset, as they are only used for inference during the knowledge transfer stage.

In the implementation, the data forwarded during inter-client communication includes the parameter and identifier of the proxy model. The proxy model identifier, which corresponds to the index of the client that generated it, is used to verify whether the client has completed the second stage. In particular, each client maintains a vector ($\mathbf{v}_i$) to record the access status of the proxy model. For instance, when client i receives the proxy model with identifier p, the status vector of client i will be updated, that is, $\mathbf{v}_{i,p} = 1$. The client i completes the second stage when all element values in $\mathbf{v}_i$ are configured to 1 (Line 8 in Algorithm 1).

Remarks. Ring-Distill opts for a ring-topology-based communication structure and can significantly reduce communication overhead among clients. In the training procedure with N participating clients, each client participates in collaborative training for only N communication rounds. In fact, other communication structures [27] and mechanisms, such as Gossip and AllReduce, can also be combined with proxy models for DFD training. However, these structures and mechanisms may have some limitations: the communication data volume per client in Gossip may increase as the number of global epochs grows, and AllReduce may lead to a communication bottleneck for individual clients.

Algorithm 1 The Key Steps of Ring-Distill

1: **Input:** $\mathcal{D}_{pri}^{(i)}, \mathbf{v}_i = \mathbf{0}, i \in [0, N)$
2: **Output:** $w_i, i \in [0, N)$
3: **Local Sparse Training Stage:**
4: For each client i: $w_i \leftarrow local\_training(w_i, \mathcal{D}_{pri}^{(i)})$
5: For each client i: $\hat{w}_i \leftarrow$ **Algorithm 2**
6: **Knowledge Transfer Stage:**
7: **for** clients $i \in [0, N)$ **in parallel do**
8: **while** $\|\mathbf{v}_i\|_0! = N$ **do**
9: Send $\hat{w}_i$ to downstream neighbor (client $i+1$)
10: Receive $\hat{w}_{i-1}$ from upstream neighbor (client $i-1$)
11: $p \leftarrow get\_identifier(\hat{w}_{i-1})$
12: $\mathbf{v}_{i,p} = 1$
13: Perform distillation by **Algorithm 3**
14: Configure $\hat{w}_{i-1}$ as $\hat{w}_i$
15: **end while**
16: **end for**

B. Privacy-Oriented Automatic Model Pruning

Ring-Distill prevents privacy leakage from clients' local models by transmitting proxy models instead of local models during the knowledge transfer stage. One of the main challenges of Ring-Distill is how to prune local models to obtain proxy models that can effectively keep both privacy and accuracy. Based on Observations 1 and 2 in Section III,

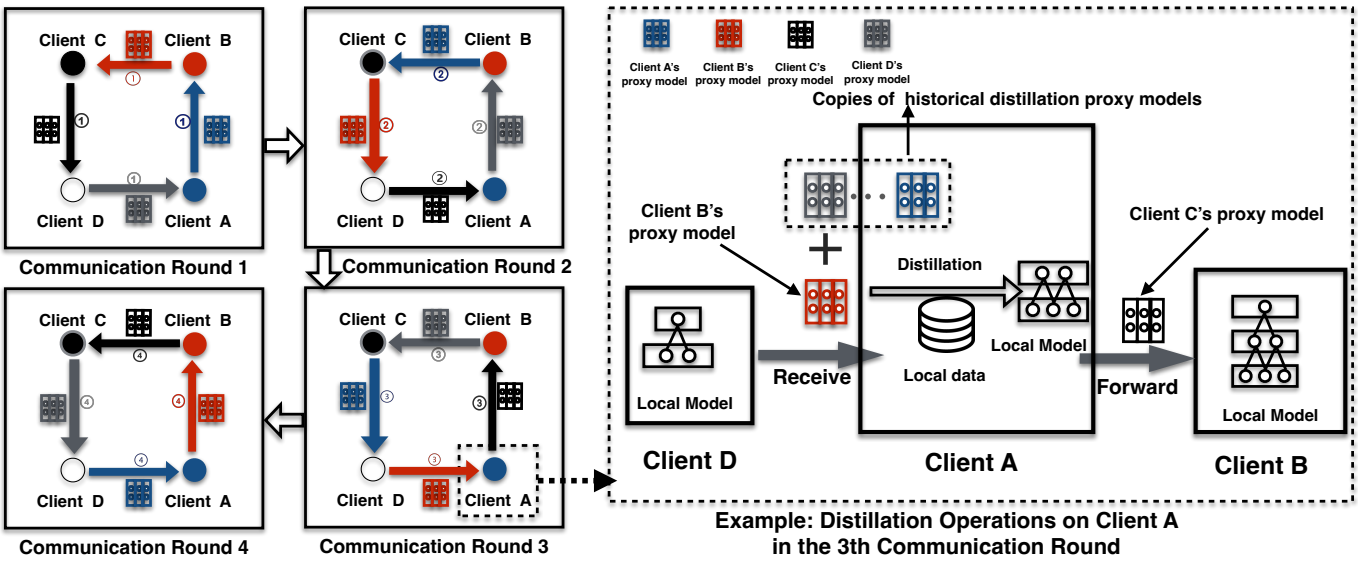


Fig. 3. The overview of Ring-Distll. The above figure demonstrates the communication procedure of proxy models among 4 clients. The entire procedure only requires 4 global epochs. Besides, the illustration within the dashed box on the right depicts the distillation procedure (HMD mechanism) of Client A.

we believe that the compressed model can effectively prevent data reconstruction attacks but may fail to prevent MIAs.

To enhance the ability to resist MIAs, Ring-Distill provides PAMP that automatically prunes the local model to create the proxy model during sparse training under the constraint of sparse training loss and inference gain. PAMP can be defined as an optimization problem:

$$\min_{m} \mathcal{L}_{loss}(w \odot m) + \lambda \mathcal{L}_{gain}(w \odot m) \tag{1}$$

$$s.t. \|m\|_0 = k, \tag{2}$$

$$m \in [0,1]^d, \tag{3}$$

where w denotes the parameter of the converged local model and is fixed during optimization. m represents the trainable mask of the same shape as w, containing d elements. Each element has a value of either 1 or 0, determining whether the element at the same index in w is pruned, i.e., the element value of 0 indicates pruning. $\odot$ denotes the element-wise pointwise multiplication operation. λ and k denote hyperparameters used to adjust contribution and compression rate, respectively. The first term in Eq. (1) represents a task-dependent loss function such as the cross-entropy loss function, and lower $\mathcal{L}_{loss}$ indicates higher accuracy of the proxy model. The second term in Eq. (1) represents the inference gain. Inference gain [28] is used to measure the benefit of MIAs. Lower $\mathcal{L}_{gain}$ indicates better defense of the proxy model against MIAs. To sum up, the goal of Eq. (1) is to reduce the inference gain while improving the accuracy of the proxy model. More specifically, $\mathcal{L}_{loss}$ can be defined in Eq. (4):

$$\mathcal{L}_{loss}(m) := \mathbb{E}_{(x,y)\sim\mathcal{D}_{pri}}[\ell_{ce}(f(w \odot m, x), y)], \tag{4}$$

where $\mathcal{D}_{pri}$ and ℓ_{ce} denote the private data of the client and the cross-entropy loss function, respectively. $f(w \odot m, x)$ can be seen as the output of the proxy model during sparse training.

Compared with $\mathcal{L}_{loss}$, the computation of $\mathcal{L}_{gain}$ is more complex since it requires an additional estimated function to measure the model's ability to resist MIAs. As literature [28], [29], $\mathcal{L}_{gain}$ can be defined as follows:

$$\begin{aligned}\mathcal{L}_{gain}(m) := \mathbb{E}_{(x,y)\sim\mathcal{D}_{pri}}[log(f_{mia}(x,y,f(w \odot m)))] \\ + \mathbb{E}_{(x,y)\sim\backslash\mathcal{D}_{pri}}[1 - log(f_{mia}(x,y,f(w \odot m)))],\end{aligned} \tag{5}$$

where $f_{mia}(\cdot)$ denotes estimated function. In practice, a work-well membership inference model can act as $f_{mia}(\cdot)$ and is used to evaluate the ability to protect privacy. Thus, $\mathcal{L}_{gain}$, i.e., Eq. (5) can be considered that evaluating the correctness of the MIA model when the target data record is sampled from the training set ($\mathcal{D}_{pri}$), or outside training set ($\backslash\mathcal{D}_{pri}$).

PAMP trains and updates the mask to automatically prune the local model via stochastic gradient descent (SGD) as specified by Eq. (1). To better train the mask, $m = [m_0, m_1, ..., m_d]$ in Eq. (1) is actually a floating-point tensor, which is then converted into a binary tensor through the operator called $Binary(\cdot)$. The operation of $Binary(\cdot)$ on each element (m_i) of the mask is defined as follows:

$$Binary(m_i) := \begin{cases} 1, & if\ m_i > \tau \\ 0, & otherwise, \end{cases} \tag{6}$$

$$\tau := Topk(m,k), \tag{7}$$

where τ denotes the top k largest element values in m. After operations of $Binary(\cdot)$, elements in m are set to 1 if their values are larger than τ, and set to 0 otherwise. However, during the training optimization process, the operator ($Binary(\cdot)$) is non-differentiable. Therefore, we adopt the Straight-Through Estimator (STE) technique [30] to obtain an approximate gradient for $Binary(\cdot)$ during backpropagation.

According to the definition of optimization provided above (i.e., Eq. (1)), during sparse training with PAMP, the output of the local model for each data sample is used not only to compute $\mathcal{L}_{loss}$ but also to feed the given MIA model for computing $\mathcal{L}_{gain}$. As a result, the mask (m) is updated based on corresponding gradients after backpropagation.

The performance of the given MIA model impacts the effectiveness of PAMP. However, a trained MIA model with high inference gain may be not accessed easily. To obtain an effective MIA model, PAMP trains the MIA model by min-max objective function since the pruned local model's performance impacts the MIA model's accuracy. Therefore, The improved version of the optimization process is defined as follows:

$$\underbrace{\min_{m}(\mathcal{L}_{loss} + \lambda \underbrace{\max_{w_{mia}} \mathcal{L}_{gain}}_{optimize\ MIA\ model})}_{optimize\ mask\ for\ privacy\ and\ pruning}. \tag{8}$$

Compared with the basic version of the optimization process (Eq. (1)), the improved version not only needs to optimize the mask based on the discriminative power of the MIA model, but also optimize the given MIA model with the help of the local model. Algorithm 2 displays key steps of PAMP mechanism. Similar to the training of GAN, PAMP alternately trains the MIA model and the local model until a balance is reached between the two. At this point, the local model will achieve an optimal balance between accuracy and protection capability. Once PAMP has completed, the MIA model will be discarded to reduce storage space. By adopting PAMP, Ring-Distill can achieve proxy models for the second stage.

Algorithm 2 Privacy-Oriented Automatic Model Pruning

1: **Require:** $\mathcal{D}_{pri}^{(i)}$, $\setminus\mathcal{D}_{pri}$, w_i, w_{mia}, m, k, T_{sparse}
2: **Output:** m
3: Initialize all elements of m to 1
4: **for** $t = 1$ to T_{sparse} **do**
5: Random select samples (x_{pri}, y_{pri}) from $\mathcal{D}_{pri}^{(i)}$
6: Random select samples (x_{out}, y_{out}) from $\setminus\mathcal{D}_{pri}$
7: Computes $\mathcal{L}_{gain}$ and update the parameters of MIA model (w_{mia}) by Eq. (5)
8: Random select samples (x_{pri}, y_{pri}) again from $\mathcal{D}_{pri}^{(i)}$
9: Compute $\mathcal{L}_{loss}$ and update the parameters of local model (w) by Eq. (4) and Eq. (6)
10: **end for**

Remarks. The proxy model generated by PAMP helps Ring-Distill to reduce communication overhead and mitigate the risk of privacy leakage. Though existing model compression techniques can effectively generate smaller models for distillation, these pruning algorithms do not further address the privacy concerns of compressed models, which may be limited in privacy-sensitive scenarios. However, without the privacy-sensitive requirement, other techniques can be applied in Ring-Distill according to application configurations in practice.

C. Historical Model-Based Distillation

In the first communication epoch, each client aligns the output of its local model with the output of the received proxy model on the private dataset, so as to transfer knowledge from the proxy model to the local model. Specifically, each local model (w_i) is additionally updated by gradients regarding the loss of distillation ($\mathcal{L}_{kd}(\cdot)$) with the proxy model ($\hat{w}_{i-1}$) received from the upstream client:

$$\mathcal{L}_{kd}(w_i^{(0)}) := \mathbb{E}_{(x)\sim\mathcal{D}_{pri}}[\ell_{kd}(f(w_i^{(0)}, x), f(\hat{w}_{i-1}^{(0)}, x))], \tag{9}$$

$$w_i^{(1)} := w_i^{(0)} - \eta\nabla_{w_i^{(0)}}\mathcal{L}_{kd}, \tag{10}$$

where $x \in \mathcal{D}_{pri}$ denotes private samples. $\ell_{kd}(\cdot)$ denotes the KL divergence loss function. After the distillation process completes, the proxy model ($\hat{w}_{i-1}^{(0)}$) is stored locally for subsequent distillation operations in the next communication rounds. In subsequent communication rounds, each client utilizes not only the received proxy model but also the previously stored historical proxy models to collectively distill knowledge into its local model. The operations of model updating via the HMD mechanism can be defined as follows:

$$\begin{aligned}\mathcal{L}(w) &:= \mathbb{E}_{(x,y)\sim\mathcal{D}_{pri}}[\ell_{ce}(f(w,x),y)] \\ &+\mathbb{E}_{(x)\sim\mathcal{D}_{pri}}[\ell_{kd}(f(w,x), \frac{1}{N_q}\sum_{j=1}^{N_q} f(\hat{w}_j, x))],\end{aligned} \tag{11}$$

where N_q and $\hat{w}_j (j \in [1, N_q])$ denote the number and parameters of historical proxy models stored locally, respectively.

Maintaining several historical proxy models may incur additional storage costs. To reduce storage costs, Ring-Distill provides two optional strategies to reduce storage costs: 1) discarding similar historical proxy models and 2) configuring the maximum storage limit. In the former, Ring-Distill evaluates the similarity between the newly received proxy model and all historical proxy models by calculating the similarity of their corresponding logits obtained on private data. If the similarity score is below the given threshold (κ_s), Ring-Distill stores the newly received proxy model (Lines 4-10 in Algorithm 3). In the implementation, some similarity calculation methods, such as cosine similarity, can be used to compute the similarity between logits. In the latter, each client maintains a queue ($\mathcal{Q}$) to store historical proxy models. When a newly proxy model is received and the queue is full, i.e., the length of the client's queue (N_q) exceeds the given threshold (κ), the earliest historical proxy model is removed based on the "first in, first out" principle (Lines 11-13 in Algorithm 3). In practice, HMD can be configured to use either strategy or both, depending on application requirements.

Remarks. We believe that the effectiveness of HMD can be attributed to two main factors. On one hand, existing works [31], [32] convey that KD works well only if the teacher model[4] generalizes better than the student model. From the perspective of ensemble learning, the average prediction obtained from several historical proxy models can be considered as the output of the teacher model that has better generalization capability. Thus, the local model can be beneficial for aligning average predictions. On the other hand, from the perspective of continual learning, distillation based on historical proxy models contributes to mitigating the issue of model forgetting. In general, model forgetting refers to the phenomenon where a model may forget previously learned

[4]In this paper, the proxy model acts as the teacher model while the local model acts as the student model.

knowledge while learning new knowledge, which can lead to a degradation in model performance. HMD ensures that, during each global epoch, the local model can "review" old knowledge from historical proxy models while learning new knowledge from the newly received proxy model.

D. Client Dropout Handling

Distributed training with the ring-topology-based communication mechanism may face robustness issues. For instance, if a client goes offline, data transmission will be interrupted, rendering distributed algorithms unable to work. To address the issue of client dropout, Ring-Distill leverages historical proxy models used in HMD. Specifically, before sending the proxy model to the downstream neighbor, the client first verifies whether its downstream client is online. Existing detection techniques in distributed systems, such as sending probe messages (which incur negligible overhead), can be integrated into Ring-Distill. If the downstream neighbor goes offline, the client searches for the next downstream client along the ring-topology-based communication structure until it finds an online client. Subsequently, the client sends the proxy model along with historical proxy models to the downstream client it finds. The found downstream client chooses which proxy model to use for distillation operations according to its training status($\mathbf{v}_i$ in Algorithm 1). In the implementation, Ring-Distill can adjust the tolerance level by tuning a hyperparameter that dictates the maximum allowable number of offline clients within the DFD system. In real-world applications, we can configure the hyperparameter based on specific requirements. For instance, in collaborative training across organizations, the physical machines and network of the participants are typically reliable, indicating that the FD system experiences minimal client dropout. As a result, the aforementioned hyperparameter can be set to zero.

Algorithm 3 Historical Model-Based Distillation (HMD)

Require: $\mathcal{D}_{pri}^{(i)}, \mathcal{Q}_i, w_i, \hat{w}_j, N_q, \kappa, \kappa_s$ ▷ $\mathcal{Q}_i$ denotes the queue that stores copies of historical proxy models
Output: w_i
Use $\hat{w}_j$ and historical proxy models $\hat{w}_{local}$ ($\hat{w}_{local} \in \mathcal{Q}_i$) for distillation by Eq.(11)
for $\hat{w}_{local} \in \mathcal{Q}_i$ **do**
  score$\leftarrow compute_similarity(\hat{w}, \hat{w}_{local}, \mathcal{D}_{pri}^{(i)})$
  if score $< \kappa_s$ **then**
    $\mathcal{Q}_i.push(\hat{w})$ ▷ push $\hat{w}$ into the queue.
    Break
  end if
end for
if $N_q == \kappa$ **then**
  $\mathcal{Q}_i.pop()$ ▷ remove the earliest historical model.
end if

V. Theoretical Analysis

The core of Ring-Distill lies in transferring knowledge among clients by proxy models instead of relying on clients' local models, which may raise the question: is it feasible for proxy models to enable knowledge transfer during federated training?

Similar to the analytical framework in the literature [31], [32], we analyze the feasibility and effectiveness of Ring-Distill with the help of VC theory. We conclude that the local model can learn other local models' knowledge from the proxy model, i.e., the proxy model can improve the generalization ability of local models.

Let $R(\cdot)$, $R_n(\cdot)$ and ϵ denote generalization error, training error over n samples, and approximation error associated with training, respectively. Let $\mathcal{O}(\cdot)$ denote estimation used to bound error in an asymptotic regime. Let $\mathcal{F}$ denote the function class and its capacity can be measured by VC-dimension. Function class with a large VC-dimension indicates that it has a large capacity and needs to learn patterns from large amounts of samples.

***Assumption** 1:* For $f_A \in \mathcal{F}_A$ with finite d_A VC-dimension and $f_B \in \mathcal{F}_B$ with finite d_B VC-dimension, we have:

$$R(f_B) - R(f_A) \geq 0, \tag{12}$$

where f_A denotes the local model that expects to transfer knowledge to another local model (f_B) in a FD system.

Assumption 1 indicates that KD is feasible for two local models in the FD system, and the student model (f_B) can benefit from KD. According to VC theory, we give the following Theorem 1:

***Theorem** 1:* Let $f_a \in \mathcal{F}_a$ with finite d_a VC-dimension denote proxy model derived from f_A. If Assumption 1 holds and $d_a \leq d_A$, under Ring-Distill setting, the generalization error of f_B can be upper bounded by $\mathcal{O}(\frac{\sqrt{d_B \log \frac{n}{d_B}} + \sqrt{d_a \log \frac{n}{d_a}}}{n^{\frac{1}{2}}})$. In particular, f_B transferring knowledge from f_a can achieve comparable performance to f_B transferring knowledge from f_A.

***Proof** 1:* Consider a binary classifier f belonging to a function class $\mathcal{F}$ with finite d VC-dimension. If $n \gg d$ and $0 < \delta < 1$, then, with the $1-\delta$ probability, the expected error of $R(f)$ is upper bounded by:

$$R(f) \leq R_n(f) + \mathcal{O}(\frac{\sqrt{d \log \frac{n}{d}}}{n^{\frac{1}{2}}}). \tag{13}$$

Let $f_T \in \mathcal{F}_T$ with d_T VC-dimension denote the real target function that trained classifiers approximate. Let f_A approach f_T, that is, client A trains its local model f_A over n samples so as to reach an acceptable convergence point. According to Eq. (13), we can give the upper bound of the generalization error of f_A:

$$R(f_A) - R(f_T) \leq \mathcal{O}(\frac{\sqrt{d_A \log \frac{n}{d_A}} - \sqrt{d_T \log \frac{n}{d_T}}}{n^{\frac{1}{2}}}) + \epsilon_{AT} \tag{14}$$

$$\leq \mathcal{O}(\frac{\sqrt{d_A \log \frac{n}{d_A}}}{n^{\frac{1}{2}}}) + \epsilon_{AT}, \tag{15}$$

where $\epsilon_{AT} = R_n(f_A) - R_n(f_T)$ is the approximation error related to f_A in the training procedure. Note that, since f_T is an ideal function, its training error over n samples $R_n(f_T) < R_n(f_A)$.

In the same way, we can give the upper bound of the generalization error of the proxy model (f_a) belonging to f_A when let f_a approach f_T:

$$R(f_a) - R(f_T) \leq \mathcal{O}(\frac{\sqrt{d_a \log \frac{n}{d_a}}}{n^{\frac{1}{2}}}) + \epsilon_{aT}, \tag{16}$$

where $\epsilon_{aT} = R_n(f_a) - R_n(f_T)$ is the approximate error related to f_a.

Under **Assumption 1** holds, f_B can effectively transfer knowledge from f_A over n samples. Therefore:

$$0 \leq R(f_B) - R(f_A) \leq \mathcal{O}(\frac{\sqrt{d_B \log \frac{n}{d_B}}}{n^{\frac{1}{2}}}) + \epsilon_{BA}, \tag{17}$$

where $\epsilon_{BA} = R_n(f_B) - R_n(f_A)$ is the approximation error.

We give the upper bound of the generalization error of f_B when f_B transfers knowledge from f_A:

$$\begin{aligned} R_{A\to B} = \\ R(f_B) - R(f_T) = R(f_B) - R(f_A) + R(f_A) - R(f_T) \\ \leq \mathcal{O}(\frac{\sqrt{d_A \log \frac{n}{d_A}} + \sqrt{d_B \log \frac{n}{d_B}}}{n^{\frac{1}{2}}}) + \epsilon_{A\to B}, \end{aligned} \tag{18}$$

where $\epsilon_{A\to B} = \epsilon_{AT} + \epsilon_{BA}$.

In the same way, we give the upper bound of the generalization error of f_B when f_B transfers knowledge from f_a:

$$\begin{aligned} R_{a\to B} = \\ R(f_B) - R(f_T) = R(f_B) - R(f_a) + R(f_a) - R(f_T) \\ \leq \mathcal{O}(\frac{\sqrt{d_B \log \frac{n}{d_B}} + \sqrt{d_a \log \frac{n}{d_a}}}{n^{\frac{1}{2}}}) + \epsilon_{a\to B}, \end{aligned} \tag{19}$$

where $\epsilon_{a\to B} = \epsilon_{aT} + \epsilon_{Ba}$ and $\epsilon_{Ba} = R_n(f_B) - R_n(f_a)$. Note that, under Ring-Distill setting, f_a is actually a compressed version of f_A. The compression operation is essentially a regularization of the model parameters, which constrains values of some elements to be set to zero from a tensor perspective. As a result, $f_a \in \mathcal{F}_a$, $\mathcal{F}_a \sqsubseteq \mathcal{F}_A$. In particular, $d_a \leq d_A$.

Theorem 1 is proved. In addition, Eq. (20) holds if $\epsilon_{A\to B}$ and $\epsilon_{a\to B}$ are sufficiently small, or both values are close, which means f_A and f_a have similar performance, i.e., both have similar training error over n samples.

$$\begin{aligned} \mathcal{O}(\frac{\sqrt{d_A \log \frac{n}{d_A}} + \sqrt{d_B \log \frac{n}{d_B}}}{n^{\frac{1}{2}}}) + \epsilon_{A\to B} \\ \geq \mathcal{O}(\frac{\sqrt{d_B \log \frac{n}{d_B}} + \sqrt{d_a \log \frac{n}{d_a}}}{n^{\frac{1}{2}}}) + \epsilon_{a\to B}. \end{aligned} \tag{20}$$

As in literature [31], [32], Actually Theorem 1 provides the upper bounds in an asymptotic regime instead of actual performance. However, Theorem 1 ensures that local models under the Ring-Distill setting can achieve a comparable generalization performance to that trained by other DFDs that transfer knowledge based on local models.

VI. Experiments

A. Experimental Setting

We build DFD system with different physical machines acting as clients (and the server). We ensure that the network environment is stable and the bandwidth is sufficient. Besides, as in literature [33], we adopt their scheme to distribute data samples into several physical machines to simulate the distribution of data in the real world (Non-IID setting). For experiments regarding heterogeneous model training, we design several groups of simulations that allow clients to train different models: in CIFAR-10 and UTKFace experiments, 1/3 of clients train ResNet-18, 1/3 of clients train ResNet-34, and the rest train VGGNet-16. In MNIST and Fashion-MNIST experiments, half of the clients train CNN and MLP, respectively. All approaches including Ring-Distill are implemented by PyTorch[5]. Each experimental result is the average obtained from 5 experiments under different random seeds.

Datasets. Several datasets widely used in literature are used in our experiments: MNIST, Fashion-MNIST, CIFAR-10, and UTKFace[6]. Specifically, UTKFace is a large-scale facial dataset comprising over 20,000 images annotated with age, gender, and ethnicity attributes. In this work, UTKFace is employed for the ethnicity classification task. As in the literature [5], for baselines that require a public dataset, some training data are sampled and serve as the public dataset. Since some FD algorithms that require a public dataset are actually beneficial from two datasets (local data and public data), for fairness, we migrate the "public dataset" into local data for baselines that do not require a public dataset.

Compared Approaches. Several representative DFDs are chosen as baselines, including CMFD [5], Def-KT [8], FedDCM [6], DCCR [24], DFLStar [25], and MUFTI [22]. Additionally, One-sided Training (standalone training without FD settings) serves as the lower-bound performance baseline in our experiments. Among these baselines, CMFD and DCCR are two logits-based decentralized FDs that require a public dataset. Def-KT, FedDCM, and DFLStar are three effective data-free decentralized FD algorithms that allow models to directly transfer knowledge to others. In particular, Def-KT does not support heterogeneous model training due to its design. In our experiments, we made some modifications to Def-KT to enable it to support heterogeneous model training. For MUFTI, we selectively employ only the first-stage algorithm of MUFTI, as its multi-task oriented second-stage algorithm is not applicable to our scenario.

Models. Five representative DNN models, including ResNet-18, ResNet-34, VGGNet-16, CNN and MLP, are trained via different DFD approaches. In particular, the dimension of 3-layers MLP is reduced from 512 to 128, and then to 10. The shape of the convolution filter in CNN is (5x5). In addition, CNN and MLP are used in experiments regarding MNIST and Fashion-MNIST datasets, while ResNet-18, ResNet-34 and VGGNet-16 are used in experiments regarding CIFAR-10 and UTKFace.

[5]https://pytorch.org

[6]https://susanqq.github.io/UTKFace/

Hyperparameters. Hyperparameter settings regarding experiments are listed in Table II. Many hyperparameter settings are the same as those in existing literature, except for the following hyperparameter: the number of communication rounds, i.e., epochs, is set differently for different datasets. However, for the proposed algorithm, the number of communication rounds is only related to the number of participating clients. In addition, during the local training stage of Ring-Distill, models are trained over 50 iterations on CIFAR-10 and 20 iterations on MNIST(and Fashion-MNIST), respectively. However, for fairness, we ensure that the total number of iterations (local epochs + communication epochs) of models is the same for all baselines. Note that, in Ring-Distill, the iterations of sparse training are used to generate proxy models rather than to update the local models. We adopt an early stopping strategy during sparse training, meaning Ring-Distill stops the sparse training once the proxy model meets certain criteria.

Attack Simulation. In our experiments, we employ three prevalent privacy attacks—namely, Membership Inference Attacks (MIA), Data Reconstruction Attacks (DRA), and Attribute Inference Attacks (AIA)—to evaluate the privacy-preserving capabilities of our algorithm. For the membership inference attack, the attack environment settings including the MIA model and data settings introduced in [28], [34] are used in our experiments. Specifically, the MIA model consists of three fully connected sub-networks: probability stream (layer size of [number of classes, 1024, 512, 64]), label stream (layer size of [number of classes, 512, 64]), and fusion stream (layer size of [256,64,1]). The fusion stream operates on the concatenation of the output of probability and label stream. For the data setup, the training dataset is divided into member training and test sets, while the test dataset is split into non-member training and test sets. Note that the non-member training samples are exclusively used for the MIA model and are not utilized for training the local models. For the data reconstruction attack, we implement the popular and effective attack method called DLG (Deep Leakage from Gradients) [35]. DLG enables adversaries to reconstruct original training data by matching and aligning gradients, leveraging stolen client-shared local models and a small set of gradients. For the attribute inference attack, we adopt the method described in [36]: an attack model is trained using internal features extracted from the target model (i.e., the victim). The attack model is subsequently leveraged to infer partial sensitive attributes of the training samples.

Metrics. We adopt several metrics to evaluate the performance of the proxy model. For experiments involving membership inference attacks, we utilize TM-score [34] in our experiments. TM-score aims to directly evaluate the performance-safety tradeoff, and it is defined as follows:

$$TM - score = \frac{(Accuracy)^{\gamma}}{MIA - Accuracy}, \tag{21}$$

where $Accuracy$ and $MIA-Accuracy$ denote the accuracy of the proxy model and MIA model (against the proxy model), respectively. The hyperparameter γ is set to 1 in our experiments. Note that, MIA can be modeled as a binary classification task, and the attack model's accuracy (MIA-Accuracy) close to 50% can be considered as random guessing for a binary classification task, indicating that the attack model does not work effectively. For experiments involving data reconstruction attacks, Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM) are used for evaluation. Specifically, MSE quantifies the average squared difference between pixel values of the original and processed images, serving as a fundamental pixel-level accuracy metric but lacking perceptual relevance to human vision. PSNR, derived from MSE, measures the logarithmic ratio of the maximum possible signal power to the noise power, with higher dB values indicating superior reconstruction fidelity. SSIM evaluates perceptual quality by comparing luminance, contrast, and structural patterns between images, yielding a score between 0 and 1 where values closer to 1 denote near-identical structural content.

TABLE II
HYPERPARAMETER SETTINGS

Hyperparameter	Value
The size of mini-batch during local training	128
The size of mini-batch during distillation	128
Learning rate	0.1
Weight decay	1e-5
Global epochs for MNIST	100
Global epochs for Fashion-MNIST	100
Global epochs for CIFAR-10	300
Concentration parameter (α_{dir}) for Dirichlet distribution (regarding non-IID settings)	0.5

B. Evaluation of PAMP mechanism

After configuring k, the PAMP mechanism can automatically prune the local model by updating the mask over several iterations of sparse training, generating a proxy model that balances privacy and accuracy. In our experiments, k is configured as half of the number of the local model's parameters, i.e., the compression rate of the proxy model is 0.5 in our experiments. Our experimental results show that the proxy model with a 0.5 compression rate can achieve similar accuracy to that of the local model. In practice, how to configure k depends on applications' requirements. To evaluate the effectiveness of PAMP in enhancing the privacy protection capabilities of the proxy model, we conduct three distinct types of privacy attacks to exfiltrate data privacy from proxy models:

Membership Inference Attack. We design experiments to compare the performance between PAMP and PAMP without the privacy constraint (i.e., without $\mathcal{L}_{gain}$ of Eq.(1)). Specifically, we train the local model (i.e., victim) on the privacy dataset until the model converges. Then, we apply two compression techniques, i.e., PAMP and PAMP without privacy constraint (namely Baseline in figures), to compress the trained model, respectively. Subsequently, the MIA model is used to attack proxy models to measure privacy protection. Table III lists experimental results. The MIA-Acc metric reflects the success rate of the attack model. Thus, lower values for proxy models are desirable. The TM-score for the model represents a comprehensive metric evaluating both model accuracy and privacy-preserving capability, where higher values indicate better performance. From Table III, we observe that, on four tasks, proxy models with varying

compression levels consistently outperform Baseline in both MIA-Accuracy and TM-score metrics. In select experimental configurations, the proxy model achieves >10% reduction in MIA success rates compared to Baseline, demonstrating significant privacy enhancement.

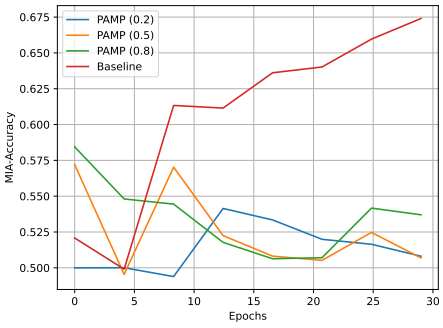
TABLE III
EXPERIMENTAL RESULTS FOR MEMBERSHIP INFERENCE ATTACKS

Tasks	Metrics	Baseline	PAMP(0.2)	PAMP(0.5)	PAMP(0.8)
MNIST (MLP)	MIA-Acc	60.25%	52.55%	50.01%	51.09%
	TM-score	1.63	1.85	1.85	1.87
Fashion-MNIST (CNN)	MIA-Acc	60.62%	47.06%	46.50%	41.78%
	TM-score	1.48	1.53	1.67	1.72
CIFAR-10 (ResNet-18)	MIA-Acc	67.40%	50.80 %	50.69 %	53.70%
	TM-score	1.35	1.75	1.80	1.72
UTKFace (VGGNet-16)	MIA-Acc	53.57%	47.06%	46.50%	41.78%
	TM-score	1.56	1.66	1.64	1.74

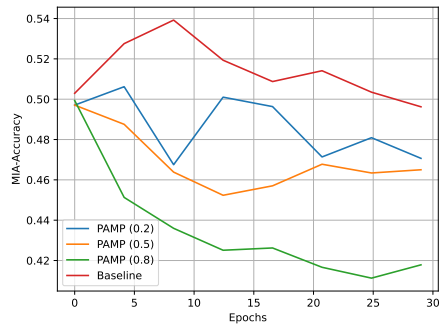
Figure 4 and 5 illustrate the performance dynamics of proxy models on the validation data throughout PAMP process, enabling visual analysis of the PAMP min-max optimization. As shown in Figure 4, MIA-Accuracy regarding PAMP (blue, orange, and green lines) is quite low and close to 50%, while MIA-Accuracy regarding baseline (red line) is larger than 50% (specifically larger than 60% in Figure 4(a)). In particular, we observe that, proxy models in Figure 4(a) and 5(a) exhibit lower performance than the baseline when the iteration count is fewer than 6 epochs. We posit that, since the PAMP-based process follows a min-max optimization framework where the proxy model seeks to minimize the inference gain loss (reducing MIA success rate) while the adversarial MIA model aims to maximize this loss (increasing MIA accuracy), the min-max optimization results in oscillating privacy-preserving capabilities of the proxy model during sparse training. Consequently, during the PAMP-based sparse training, the proxy model occasionally exhibits performance degradation in individual epochs—even falling below the baseline in terms of privacy protection. However, as the number of iterations in sparse training increases, the MIA-Accuracy of PAMP tends to decrease and stabilize, and proxy models finally achieve favorable performance under different compression rates.

From the perspective of the accuracy-privacy trade-off, Figure 5 shows the proxy model generated by PAMP achieves a higher TM-score, indicating that the proxy model generated by PAMP can maintain both accuracy and privacy. In particular, as the number of iterations in sparse training increases, TM-score related to PAMP remains stable and is consistently higher than that of baseline. Additionally, we found that PAMP achieves a favorable TM-score under different parameter settings (blue, orange, and green lines in Figure 5). In our experiments, configuring compression rate as 0.5 enables the proxy model to achieve an optimal balance among precision, privacy preservation, and model size. In practice, the compression rate for PAMP in RingDistill can be adjusted to an appropriate value based on different applications and ML tasks.

Data Reconstruction Attack. DLG attacks [35] are executed on uncompressed models and proxy models with varying compression ratios across three tasks: CNN/Fashion-MNIST, ResNet-18/CIFAR-10, and VGGNet-16/UTKFace. These experiments aim to exfiltrate training data samples from target models (i.e., victims). Experimental results are summarized in



(a) ResNet-18 on CIFAR-10



(b) VGGNet-16 on UTKFace

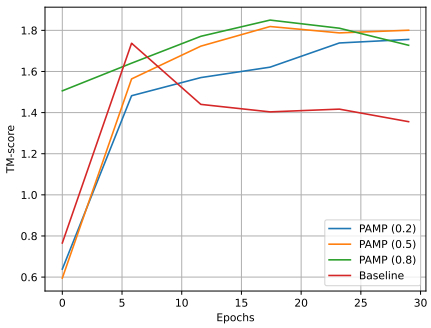
Fig. 4. MIA-Accuracy (%) comparison for proxy models generated by different settings. Lower is better. The blue, orange, and green lines denote the MIA-Accuracy of the MIA model used to attack proxy models. These proxy models are generated by PAMP mechanism with 0.2, 0.5, and 0.8 compression rates, respectively. An MIA-Accuracy close to 50% indicates that the proxy model is unable to defend against MIA attacks in MIA classification.

Table IV. Lower MSE, higher PSNR, and higher SSIM values indicate greater similarity between reconstructed and original images, indicating more successful data reconstruction attacks. In contrast, a higher MSE alongside lower PSNR and SSIM values demonstrates a stronger resistance of the target model against such attacks. Note that the values reported in Table IV are averages computed over test samples.

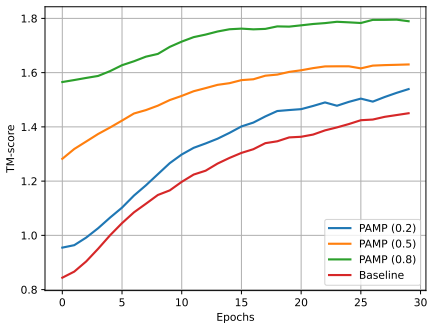
TABLE IV
EXPERIMENTAL RESULTS FOR DATA RECONSTRUCTION ATTACKS

Tasks	Metrics	Baseline	PAMP(0.2)	PAMP(0.5)	PAMP(0.8)
Fashion-MNIST (CNN)	MSE	0.043	0.169	0.11	0.128
	PSNR	13.89dB	7.88dB	9.61dB	9.31dB
	SSIM	0.827	0.181	0.290	0.144
CIFAR-10 (ResNet-18)	MSE	0.011	0.041	0.038	0.068
	PSNR	19.44dB	13.79dB	14.15dB	11.63dB
	SSIM	0.727	0.390	0.037	0.143
UTKFace (VGGNet-16)	MSE	0.036	0.073	0.060	0.041
	PSNR	14.42dB	11.32dB	11.02dB	12.4dB
	SSIM	0.51	0.303	0.415	0.372

Figure 6 depicts subsets of reconstructed Fashion-MNIST samples from both baseline and proxy models. Key observations reveal that, compared to the baseline, reconstruction attacks fail to effectively reconstruct training samples from proxy models. Additionally, we simulate scenarios where proxy models are hijacked and compromised during RingDistill training. Specifically, within the semi-honest threat model scenario, for the task training ResNet-18 on CIFAR-10, we assume an adversary has obtained one of the client's proxy models to perform data reconstruction attacks. Figure 7 exhibits partial attack outcomes. Crucially, reconstruction attacks consistently fail to effectively reconstruct training



(a) ResNet-18 on CIFAR-10



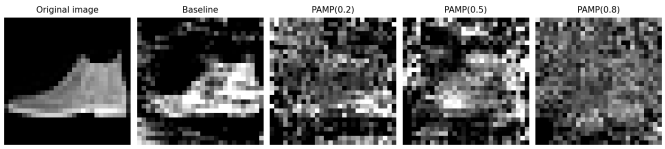
(b) VGGNet-16 on UTKFace

Fig. 5. TM-score comparison for proxy models generated by different settings. Higher is better. The blue, orange, and green lines denote the TM-score of the proxy model generated by PAMP with 0.2, 0.5, and 0.8 compression rates, respectively. The red line represents PAMP without the privacy constraint(Baseline). A high TM-score indicates that the proxy model can better balance both accuracy and privacy. Note that the curves in figures are smoothed for visually intuitive presentation.

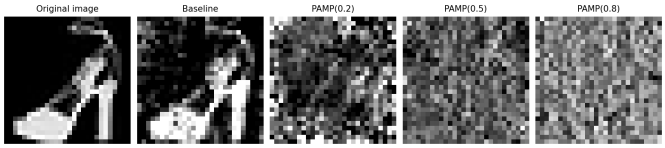
samples from proxy models across varying compression rates, demonstrating the inherent privacy resilience of PAMP.

Attribute Inference Attack. As in literature [36], we utilize the UTKFace dataset for attribute inference attacks (AIA) due to its multi-attribute annotations per image (e.g., ethnicity and gender labels). These annotations serve as sample attributes in our experiments. We conduct attribute inference attacks against target models trained for ethnicity classification on UTKFace, specifically inferring the gender attribute (i.e., gender labels) of training samples. To simulate real-world AIA scenarios where attackers possess limited data resembling training samples for training attack models, half of the data samples from the UTKFace test dataset are used to train attack models. Figure 8 demonstrates the attack success rate of AIA. Since the attack model aims to infer gender labels, AIA in our experiments can be considered as a binary classification task. When the attack success rate falls below 50% (the random guess baseline), we believe that AIA is ineffective. Our experiments validate that proxy models generated through PAMP maintain attack success rates below 51% – demonstrably close to the security threshold (50%). In comparison, AIA successfully infers gender attributes from the baseline with high accuracy (approaching 76%). The experimental results demonstrate the effectiveness of proxy models in mitigating AIAs.

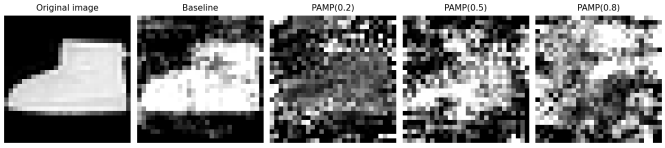
Discussion. Experimental results across multiple tasks demonstrate that the proxy model effectively mitigates three



(a) The image of sports shoes



(b) The image of high-heeled shoes



(c) The image of snow boots

Fig. 6. Visual comparison between original images and reconstructed counterparts. Within each image group, the reconstructions are arranged from left to right as follows: uncompressed model, then proxy models with compression rates of 0.2, 0.5, and 0.8.

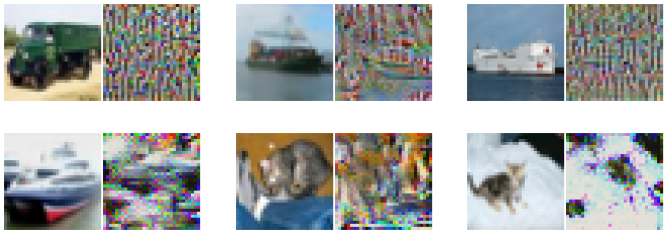


Fig. 7. Visualization of reconstruction attack results. Six groups of subfigures are presented, with the left image showing the original sample and the right image displaying the reconstructed counterpart in each pair.

mainstream privacy attacks (MIA, DRA, and AIA), thus validating the superiority of the PAMP mechanism. In addition, our observations reveal no strong correlation between proxy model compression rates and privacy protection efficacy under three attack scenarios. Specifically, higher compression levels do not consistently improve protection metrics, such as MIA Accuracy and TM-score. We posit that the PAMP mechanism decouples the strong correlation between model outputs and sensitive features during compression, rendering the attacker's predictions statistically indistinguishable from random guessing regardless of the compression level.

C. Evaluation of Historical Model-Based Distillation.

One of the advantages of HMD mechanism in Ring-Distill is to improve the effectiveness of knowledge transfer. To validate the effectiveness of HMD, we conducted experiments on four tasks (MNIST, Fashion-MNIST, CIFAR-10, and UTKFace) in both homogeneous and heterogeneous training scenarios, using different numbers of historical models (parameterized by κ). Experimental results in Table V conclusively demonstrate that HMD effectively improves the final accuracy of local models across all clients. Moreover, we observe that HMD reduces oscillations in model accuracy throughout the training process.

Ablation Result. The HMD mechanism plays an indispensable role in Ring-Distill. The performance of Ring-Distill

without HMD may be affected. In this ablation study, setting $\kappa = 1$ for the HMD mechanism corresponds to vanilla Ring-Distill without HMD capabilities, as clients never store historical proxy models locally. Experimental results presented in Table V demonstrate that, when $\kappa = 5$, compared to Ring-Distill without HMD ($\kappa = 1$), HMD significantly boosts the average accuracy of all local models: approximately +7% on MNIST, +5% on Fashion-MNIST, +4% on CIFAR-10, and +4% on UTKFace. When storage limitations are not considered, increasing the number of historical models ($\kappa >5$) may potentially enhance accuracy. From the client-level perspective, Figure 9 illustrates the training status of one client in Ring-Distill involving 10 participating clients. We observe significantly lower local model accuracy in Ring-Distill without HMD (blue line) compared to HMD-enabled variants (green and orange lines). In particular, without HMD mechanism, the performance of the local model is very unstable and degraded. For instance, local model accuracy declines during the 2nd/3rd rounds (blue line in Figure 9(a)) and the 3rd/6th rounds (Figure 9(b)), indicating catastrophic forgetting of previously acquired knowledge. In contrast, HMD-augmented variants ($\kappa > 1$) maintain stable performance during training, demonstrating HMD's efficacy in mitigating catastrophic forgetting via historical knowledge retention.

Configuration regarding κ. In general, as the number of historical models (κ) increases, the performance (accuracy and stability) of the local model improves. However, it may incur higher storage costs if κ is configured as a larger value. In practice, we can configure κ based on the trade-off between accuracy and storage costs. From the experimental results shown in Figure 9, we found that $\kappa = 3$ and $\kappa = 5$ achieve comparable accuracy (orange and green lines). Therefore, we believe that $\kappa = 3$ is an appropriate choice in our MNIST, Fashion-MNIST, CIFAR-10, and UTKFace experiments, as it offers higher benefits with lower storage costs.

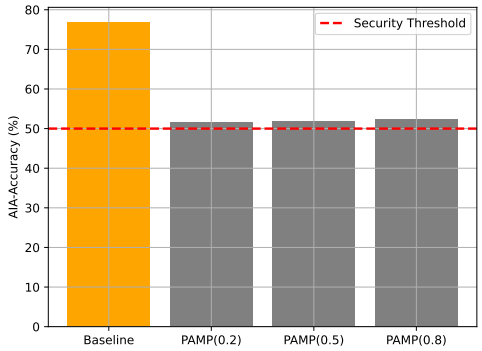
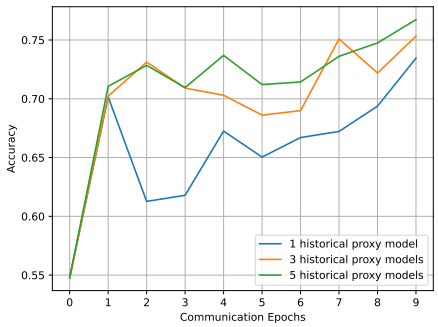


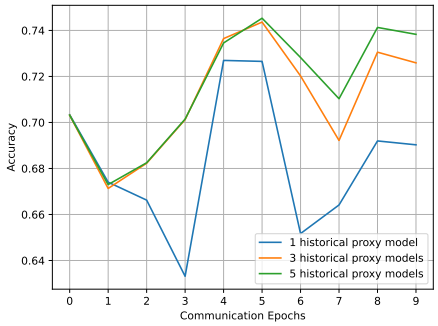
Fig. 8. Attribute Inference Attack Success Rate (AIA- Accuracy). The attack model attempts to infer gender attributes of UTKFace samples utilized in the target model's training process.

D. Performance Comparison with Baselines

Table VI records all experimental results. The maximum, minimum, and average values in Table VI record the best accuracy, the worst accuracy, and the average accuracy of local models among clients, respectively. The difference between the best and worst accuracy can reflect the severity of the Non-IID issue: the larger gap indicates DFDs may not effectively



(a) ResNet-18 on CIFAR-10



(b) VGGNet-16 on UTKFace

Fig. 9. Performance comparison with different configurations regarding the number of historical proxy models stored locally.Note that the accuracy at Epoch 0 actually indicates the final accuracy of the local model after local sparse training (the first stage).

TABLE V
ACCURACY (%) COMPARISON UNDER DIFFERENT VALUES OF κ.

Dataset	Models	κ	Maximum	Minimum	Average
MNIST	CNN	1	84.09	60.02	77.83
	MLP	3	92.98	74.01	84.90
		5	92.32	75.26	84.96
Fashion-MNIST	CNN	1	76.34	64.56	69.27
	MLP	3	80.0	65.06	74.93
		5	80.74	67.42	75.07
CIFAR-10	ResNet-18	1	78.61	68.84	70.58
		3	79.23	66.6	72.66
		5	79.92	70.67	74.82
UTKFace	ResNet-18	1	73.32	66.26	69.61
	ResNet-34	3	75.27	67.81	72.55
	VGGNet-16	5	76.24	71.01	74.16

mitigate it. Besides, the column named 'Models' in Table VI records the kind of models involved in training. Overall, Ring-Distill can outperform baselines in both heterogeneous and homogeneous model training scenarios. Also, the proposed algorithm can address the Non-IID issue better than baselines.

Heterogeneous Training. Cases 1-40 in Table VI record experimental results about heterogeneous training on MNIST, Fashion-MNIST, CIFAR-10 and UTKFace. From the results, we observe that models trained via Ring-Distill can achieve comparable or even better accuracy than baselines on all datasets. More precisely, the overall accuracy of all clients' local models (i.e., average accuracy) outperforms that of models trained with other baselines. From the results in Table VI, Ring-Distill can improve nearly 7%- 13% accuracy compared to One-sided Training, and improve nearly 5% compared to

other DFDs. Furthermore, the difference between the maximum and minimum accuracy of local models trained with Ring-Distill is smaller than that of models trained with other baselines, indicating that Ring-Distill effectively mitigates the Non-IID issue, whereas other baselines may lead to significant accuracy discrepancies among clients. Figure 10 can illustrate the observation more intuitively. Red dots in Figure 10 show that in the training system using Ring-Distill, the accuracy range for clients is from 65% to 80%, and the entire accuracy range shifts towards higher precision.

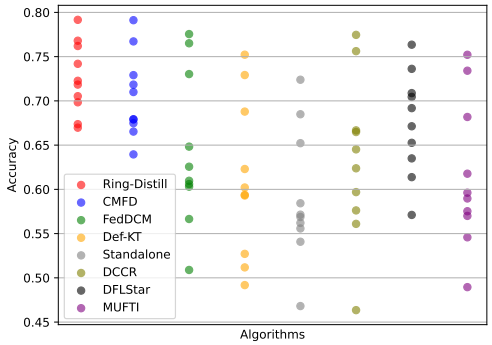


Fig. 10. Illustration of local model accuracy per client: each dot represents one client, and dots of the same color show the accuracy distribution for 10 clients trained with a specific algorithm. E.g., red dots indicate accuracy achieved with Ring-Distill. The smaller accuracy differences among local models under Ring-Distill suggest that it effectively addresses the Non-IID problem.

Homogeneous Training. Cases 41-80 in Table VI record experimental results about homogeneous training on MNIST, Fashion-MNIST, and CIFAR-10. The results in this group of experiments align well with our expectations. Compared with other model-based DFDs, the proposed algorithm performs well in homogeneous training as it does in heterogeneous training, and shows a significant advantage in terms of accuracy. In certain cases, Ring-Distill can even improve average accuracy by nearly 10% compared to some DFDs on CIFAR-10. Also, Ring-Distill outperforms baselines on MNIST and Fashion-MNIST. Furthermore, we observe that the range between the maximum and minimum performance of Ring-Distill is smaller than that of the baselines, which may indicate that the proposed algorithm may effectively address the Non-IID issue in homogeneous training. To sum up, the results regarding both heterogeneous and homogeneous training convey that, Ring-Distill achieves the expected accuracy and performance on several datasets compared to baselines.

Comparison under Diverse Non-IID Settings. As in literature [33], we employ a Dirichlet distribution with concentration parameter α_{dir}=0.5 to simulate Non-IID data settings. To validate our algorithm's robustness in diverse Non-IID settings, we conduct two sets of experiments on the Fashion-MNIST dataset using distinct Dirichlet concentration parameters (α_{dir}=0.2 and α_{dir}=0.9). As quantitatively documented in Table VII, the results demonstrate Ring-Distill's consistent effectiveness in addressing heterogeneous data distributions. Furthermore, we experimentally investigate an additional Non-IID configuration: Label Distribution Skew [37]. Specifically, we design experiments leveraging UTKFace's long-tailed ethnicity label distribution—a natural fit for simulating FD under label-imbalanced conditions. As quantitatively validated in Table VI (Cases 33-40), our approach achieves comparable accuracy under label distribution skew, demonstrating its robustness against severe class imbalance inherent in FL/FD.

Scalability. To verify the scalability of Ring-Distill in cross-silo federated training, we additionally conduct experiments in which 20 clients participate in heterogeneous training (Cases 17-24 in Table VI). Experimental results demonstrate that Ring-Distill still performs well even when the number of clients increases. It is worth mentioning that, since each client executes the second training stage in parallel, no additional waiting time is introduced for the clients when the number of clients increases. In practice, the cross-silo training system typically has fewer participants compared to cross-device training scenarios. Therefore, we believe that the scalability of the proposed algorithm is sufficient to meet the client quantity requirements in real-world, cross-silo training scenarios.

Discussion. From experimental results, we believe two factors contribute to the high accuracy under Ring-Distill: (1) clients can effectively learn knowledge from proxy models, and (2) the HMD mechanism helps mitigate model forgetting.

E. Communication Costs Comparison

We evaluate the communication cost of Ring-Distill in terms of communication frequency and data volume, respectively.

Communication Frequency. For typical DFDs, such as CMFD, the training process may require hundreds or even thousands of communication rounds, depending on the complexity of ML tasks. Different from baselines, the communication rounds per client of Ring-Distill are only related to the number of participating clients. Thus, from the perspective of clients, the proposed algorithm has a clear advantage.

Communication Data Volume. Each client only needs to send/receive a proxy model (compressed model) in each communication round, we believe that Ring-Distill can effectively reduce the amount of data by configuring an appropriate value of k according to scenarios. For example, the size of CNN is about 0.03 MB. If $k = 0.5$, the size of the proxy model is about 0.015 MB. To further evaluate the communication volume during the Ring-Distill training process, we conduct experiments in a homogeneous model training scenario. Specifically, we use different algorithms to train ResNet-18, ResNet-34, and VGGNet-16, respectively. We record the communication data

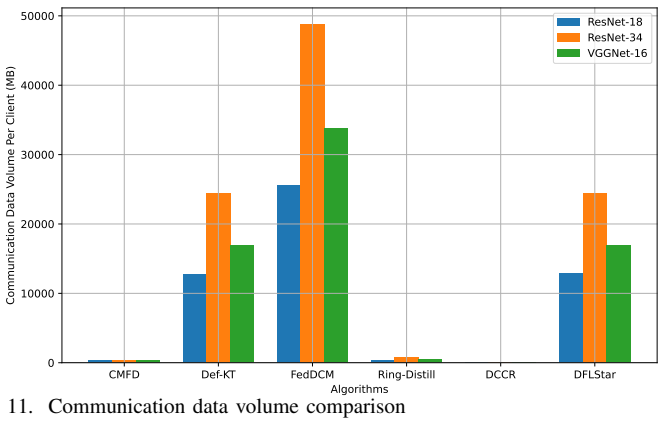


Fig. 11. Communication data volume comparison

TABLE VI
ACCURACY (%) COMPARISON UNDER DIFFERENT SETTINGS.

Cases	Datasets	Models	Approaches	Maximum	Minimum	Average
Heterogeneous Training						
1	MNIST	CNN MLP	One-sided Training	87.87	54.59	70.44
2			CMFD	89.82	58.86	80.80
3			Def-KT	87.76	56.79	75.98
4			FedDCM	87.98	58.47	79.08
5			DCCR	88.16	55.64	78.40
6			DFLStar	86.27	53.52	77.57
7			MUFTI	87.69	54.26	77.32
8			Ring-Distill	92.98	74.01	**84.90**
9	Fashion-MNIST	CNN MLP	One-sided Training	79.71	65.11	67.75
10			CMFD	82.39	67.71	74.09
11			Def-KT	79.59	60.56	70.24
12			FedDCM	79.14	62.85	70.93
13			DCCR	82.65	66.22	72.88
14			DFLStar	77.81	63.65	69.48
15			MUFTI	81.03	64.3	71.36
16			Ring-Distill	80.0	65.06	**74.93**
17	Fashion-MNIST **(20 clients)**	CNN MLP	One-sided Training	76.22	53.86	60.98
18			CMFD	79.39	58.22	72.15
19			Def-KT	72.48	50.11	62.99
20			FedDCM	74.77	51.3	65.57
21			DCCR	78.38	55.68	71.15
22			DFLStar	72.98	51.96	62.0
23			MUFTI	74.95	46.49	62.78
24			Ring-Distill	78.68	62.97	**72.91**
25	CIFAR-10	ResNet-18 ResNet-VGGNet-16	One-sided Training	72.39	46.81	59.12
26			CMFD	79.12	63.95	70.54
27			Def-KT	75.22	49.18	61.11
28			FedDCM	77.55	50.89	64.39
29			DCCR	77.46	46.35	63.29
30			DFLStar	76.35	57.11	67.49
31			MUFTI	75.21	48.95	61.52
32			Ring-Distill	79.16	66.97	**72.52**
33	UTKFace	ResNet-18 ResNet-VGGNet-16	One-sided Training	70.48	63.06	67.91
34			CMFD	74.48	68.56	72.06
35			Def-KT	72.50	67.21	71.29
36			FedDCM	74.59	69.10	72.50
37			DCCR	72.27	54.24	69.24
38			DFLStar	77.03	51.97	72.57
39			MUFTI	73.20	56.25	72.20
40			Ring-Distill	75.55	67.80	**72.55**
Homogeneous Training						
41	MNIST	MLP	One-sided Training	85.1	55.47	76.66
42			CMFD	89.5	60.31	79.62
43			Def-KT	85.56	57.56	78.47
44			FedDCM	89.19	59.93	79.49
45			DCCR	86.24	55.72	77.19
46			DFLStar	84.84	61.32	77.78
47			MUFTI	84.53	54.56	77.22
48			Ring-Distill	91.47	75.11	**84.93**
49	Fashion-MNIST	CNN	One-sided Training	74.71	62.18	70.48
50			CMFD	78.58	63.03	72.88
51			Def-KT	79.49	60.37	72.37
52			FedDCM	78.82	61.63	71.9
53			DCCR	81.84	64.1	72.11
54			DFLStar	81.46	66.46	72.2
55			MUFTI	80.09	63.11	72.46
56			Ring-Distill	79.9	63.77	**73.19**
57	CIFAR-10	VGGNet-16	One-sided Training	73.44	45.37	56.00
58			CMFD	77.78	61.55	69.16
59			Def-KT	73.1	48.3	61.09
60			FedDCM	74.74	50.1	62.14
61			DCCR	75.05	48.16	62.55
62			DFLStar	75.11	52.14	63.09
63			MUFTI	75.77	49.24	61.12
64			Ring-Distill	77.13	66.28	**72.09**
65	CIFAR-10	ResNet-18	One-sided Training	72.68	47.74	59.5
66			CMFD	79.52	63.32	70.54
67			Def-KT	74.44	51.49	61.6
68			FedDCM	77.6	54.07	65.66
69			DCCR	77.48	50.63	63.95
70			DFLStar	74.78	49.14	61.60
71			MUFTI	76.13	48.75	61.84
72			Ring-Distill	79.23	66.6	**72.66**
73	CIFAR-10	ResNet-34	One-sided Training	72.15	47.13	58.62
74			CMFD	79.57	62.2	68.08
75			Def-KT	74.75	49.81	60.20
76			FedDCM	76.43	51.45	63.64
77			DCCR	77.45	48.68	66.62
78			DFLStar	76.59	49.95	66.88
79			MUFTI	75.23	49.33	67.61
80			Ring-Distill	78.52	68.13	**72.23**

TABLE VII
CNN ACCURACY (%) COMPARISON ON FASHION-MNIST

Concentration parameter (α_{dir})	Approaches	Maximum	Minimum	Average
0.2	One-sided Training	54.82	19.85	40.40
	CMFD	55.98	27.97	43.44
	Def-KT	61.94	45.20	46.9
	FedDCM	63.55	27.78	48.67
	DCCR	68.68	27.59	51.0
	DFLStar	61.13	27.88	47.30
	MUFTI	68.59	27.99	51.48
	Ring-Distill	66.46	32.44	52.01
0.9	One-sided Training	82.66	73.1	75.69
	CMFD	84.71	74.45	80.1
	Def-KT	84.39	78.33	80.46
	FedDCM	81.82	71.9	80.10
	DCCR	84.48	73.38	80.12
	DFLStar	81.46	72.2	76.84
	MUFTI	84.09	73.03	78.34
	Ring-Distill	82.94	74.8	81.5

volume during the training process. The experimental results are shown in Figure 11. Please note that Figure 11 only records the average communication data volume of clients. The overall system's communication data volume is approximately proportional to the communication volume per client. From the experimental results, we can see that Ring-Distill has a clear advantage in terms of communication data volume. Even compared to CMFD, one of the variants of logits-based DFDs, Ring-Distill maintains a similarly low communication data volume. The communication data volume of logits-based DFDs, such as CMFD and DCCR, is related to the quantity of public data samples and the number of communication rounds, resulting in generally low communication overhead. However, a few of global epochs in Ring-Distill contribute to its satisfactory results. Specifically, experimental results show that the volume of communication data per client in Def-KT, FedDCM, and DFLStar matches the theoretical predictions. Specifically, in the same number of global epochs, the communication data volume in Def-KT is approximately half of that in FedDCM. It's important to note that in our experiments, the model type of the global model in FedDCM is the same as the type of the local model. In this experimental comparison, we intentionally excluded MUFTI as a baseline due to its fundamentally distinct centralized-server communication topology.

F. Computational Costs Comparison

To analyze Ring-Distill's training time advantage, we first evaluate the compute budget and training time of PAMP. Specifically, we benchmark PAMP-based sparse training against conventional model training (e.g., local training in FD) using identical hardware configurations (e.g., NVIDIA GeForce RTX 3090 GPU). We independently quantified two key efficiency metrics: 1) per-sample FLOPs cost, and 2) Per-batch training time with fixed batch size (128 samples). Experimental results on four benchmark tasks are comprehensively presented in Table VIII.

As evidenced by Table VIII, though PAMP incurs marginally higher per-sample computational costs compared to conventional model training, both methods demonstrate statistically indistinguishable training time when operating at identical batch sizes. This observation aligns with our analysis: compared to conventional model training, PAMP requires additional computational resources to train a lightweight MIA model. However, the MIA model maintains an extremely compact architecture (e.g., MLP classifier), resulting in negligible overhead. Furthermore, by implementing an early stopping strategy, the proxy model typically converges within 30 local iterations during PAMP-based sparse training, as empirically validated across our benchmark tasks. Also, each client executes PAMP only once throughout the entire Ring-Distill training cycle. Based on the above comprehensive analysis, we believe that PAMP-based sparse training can be functionally regarded as a variant of local training, which introduces no statistically significant training time overhead to Ring-Distill.

Regarding the training time of DFD algorithms, for clients in baseline methods such as CMFD and FedDCM, the composition of training time (t_{train}) is defined concisely as follows: $t_{train} = E \cdot (t_{local} + t_{comm} + t_{distill})$, where t_{local}, t_{comm}, and

$t_{distill}$ represent the runtime of local training, communication latency, and runtime of knowledge distillation phase, respectively. E represents the number of communication rounds (i.e., epochs). For clients in Ring-Distill, t_{train} is defined as follows: $t_{train} = t_{local} + t_{PAMP} + (N-1)(t_{comm} + t_{distill})$, where t_{PAMP} represents the runtime of PAMP. In general, given sufficient local computational resources (e.g., mainstream hardware such as NVIDIA RTX 2080/3090 GPUs), local training, knowledge distillation, and PAMP-based sparse training impose negligible overhead and do not constitute performance bottlenecks. As a result, empirical measurement indicate that the overall training time scales linearly with the number of communication rounds, as communication latency dominates. Ring-Distill restricts each client to only $N-1$ communication exchanges, where N denotes the total number of clients. Consequently, in practical deployments where communication rounds vastly outnumber participating clients—as is typical in federated scenarios—the proposed algorithm is superior to baselines in terms of training time.

TABLE VIII
COMPUTATIONAL COSTS COMPARISON

Metrics	Compute Budget (FLOPs/Sample)		Training Time (Second/Batch)	
Methods	Conventional Model Training	Sparse Training with PAMP	Conventional Model Training	Sparse Training with PAMP
MLP (MNIST)	≈ 476.4 KFLOPs	≈ 3.351 MFLOPs	≈0.03s	≈0.03s
CNN (Fashion-MNIST)	≈ 1.442 MFLOPs	≈ 4.638 MFLOPs	≈0.04s	≈0.04s
ResNet-18 (CIFAR-10)	≈ 1.674 GFLOPs	≈ 2.234 GFLOPs	≈0.26s	≈0.26s
VGGNet-16 (UTKFace)	≈ 46.203 GFLOPs	≈ 61.606 GFLOPs	≈3.29s	≈3.29s

G. Evaluation of Client Dropout Handling

Due to the HMD mechanism, Ring-Distill can address the client dropout issue. To evaluate the ability to handle client dropout issues, we designed experiments simulating random client dropout events at different time points (global epochs) during training by using varied random seeds. Note that, we assume that at most one client drops out during the training in FD system with 10 clients, as we believe that for cross-silo DFD, the resources of participating clients are relatively stable, and large-scale failures are unlikely in the system. Table IX records the experimental results. 'Client ID'in Table IX denotes ID of the disconnected client. The value in parentheses represents the final accuracy of the disconnected client. 'Point' denotes the global epoch at which the client drops out. Overall, despite the client dropouts, most of local models of clients can still maintain satisfactory accuracy (average accuracy). In particular, when the client dropout event occurs near the end of the training process, the dropout problem barely affects the performance of local models (Cases 1,2,5 in Table IX). However, when the dropout event occurs at the beginning of the training, the average accuracy of local models drops significantly. We believe that the local model of the disconnected client does not participate in the subsequent co-training, and its lower final accuracy contributes to a decrease in the average accuracy of all local models. Indeed, the final accuracy of the local model of the disconnected client may be the lowest in the system (Cases 3 and 4 in Figure IX). Thus, we conclude that, excluding the disconnected client, local models of the remaining clients still achieve satisfactory accuracy, which indicates Ring-Distill can maintain performance even if the system incurs client dropout events.

TABLE IX
ACCURACY (%) UNDER RANDOMIZED CLIENT DROPOUT

Cases	Client ID	Point	Maximum	Minimum	Average
1	4 (60.07)	8	80.96	60.07	70.99
2	9 (74.44)	9	77.09	61.74	69.33
3	6 (56.58)	2	80.72	56.58	68.68
4	9 (58.53)	2	75.98	58.53	67.45
5	2 (62.9)	7	80.67	59.85	70.02

VII. CONCLUSION

In this paper, we propose a two-stage, model-based DFD algorithm called Ring-Distill. Ring-Distill incorporates two techniques: 1) privacy-oriented automatic model pruning (PAMP) mechanism, and 2) historical model-based distillation (HMD) mechanism, to achieve the goals of efficient communication, public dataset-free distillation, and privacy protection. In addition, theoretical analysis and comprehensive experiments regarding Ring-Distill are provided. Future work will mainly focus on the following two aspects: 1) how to apply proxy models to centralized federated training scenarios? and 2) How to adapt Ring-Distill to cross-device FD scenarios?

VIII. ACKNOWLEDGMENTS

Weigang Wu (e-mail: wuweig@mail.sysu.edu.cn) and Jiannong Cao are the corresponding authors. This research was supported in part by HK RGC Collaborative Research Fund (No. C5032-23GF) HK RGC General Research Fund (No. PolyU-15228623), and Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University. Also, supported by the National Natural Science Foundation of China (No.62372487), and the Guangdong Natural Science Foundation General Project (No. 2024A1515010378).

REFERENCES

[1] F. Lyu, C. Tang, Y. Deng, T. Liu, Y. Zhang, and Y. Zhang, "A prototype-based knowledge distillation framework for heterogeneous federated learning," in *43rd IEEE International Conference on Distributed Computing Systems, (ICDCS 2023)*, 2023.

[2] J. Zhang, C. Zhu, C. Ge, C. Ma, Y. Zhao, X. Sun, and B. Chen, "Badcleaner: Defending backdoor attacks in federated learning via attention-based multi-teacher distillation," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 5, pp. 4559–4573, 2024.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, and et al., "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021.

[4] H. Duan, Z. Peng, L. Xiang, Y. Hu, and B. Li, "A verifiable and privacy-preserving federated learning training framework," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 05, pp. 5046–5058, 2024.

[5] A. Taya, T. Nishio, M. Morikura, and K. Yamamoto, "Decentralized and model-free federated learning: Consensus-based distillation in function space," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, no. 1, pp. 799–814, 2022.

[6] Y. Huang, L. Kong, Q. Li, and B. Zhang, "Decentralized federated learning via mutual knowledge distillation," in *IEEE International Conference on Multimedia and Expo, (ICME 2023)*, 2023.

[7] E. Hallaji, R. Razavi-Far, M. Saif, B. Wang, and Q. Yang, "Decentralized federated learning: A survey on security and privacy," *IEEE Transactions on Big Data*, vol. 10, no. 2, pp. 194–213, 2024.

[8] C. Li, G. Li, and P. K. Varshney, "Decentralized federated learning via mutual knowledge transfer," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1136–1147, 2022.

[9] E. Jeong and M. Kountouris, "Personalized decentralized federated learning with knowledge distillation," in *IEEE International Conference on Communications, (ICC 2023)*, 2023.

[10] K. N. Kumar, C. K. Mohan, and L. R. Cenkeramaddi, "The impact of adversarial attacks on federated learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 2672–2691, 2024.

[11] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proceedings of the 38th International Conference on Machine Learning, (ICML 2021)*, 2021.

[12] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 191–205, 2023.

[13] D. Xiao, D. Yang, J. Li, X. Chen, and W. Wu, "Privacy leakage from logits attack and its defense in federated distillation," in *54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks,(DSN 2024)*, 2024.

[14] P. Han, X. Shi, and J. Huang, "Fedal: Black-box federated knowledge distillation enabled by adversarial learning," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 11, pp. 3064–3077, 2024.

[15] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3347–3366, 2023.

[16] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," *CoRR*, vol. abs/1910.03581, 2019. [Online]. Available: https://arxiv.org/abs/1910.03581

[17] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI 2021)*, 2021.

[18] D. Sui, Y. Chen, J. Zhao, Y. Jia, Y. Xie, and W. Sun, "FedED: Federated learning via ensemble distillation for medical relation extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, (EMNLP 2020)*, 2020.

[19] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. S. Doermann, and A. Innanje, "Preserving privacy in federated learning with ensemble cross-domain knowledge distillation," in *The Thirty-Sixth AAAI Conference on Artificial Intelligence, (AAAI 2022)*, 2022.

[20] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, (NeurIPS 2020)*, 2020.

[21] Q. Li, B. He, and D. Song, "Practical one-shot federated learning for cross-silo setting," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI 2021)*, 2021.

[22] K. Gai, Z. Wang, J. Yu, and L. Zhu, "MUFTI: multi-domain distillation-based heterogeneous federated continuous learning," *IEEE Transactions on Information Forensics and Security*, vol. 20, no. 1, pp. 2721–2733, 2025.

[23] R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," in *6th International Conference on Learning Representations, (ICLR 2018)*, 2018.

[24] A. Taya, Y. Nishiyama, and K. Sezaki, "Convergence visualizer of decentralized federated distillation with reduced communication costs," in *IEEE Global Communications Conference, (GLOBECOM 2023)*, 2023.

[25] B. Soltani, V. Haghighi, Y. Zhou, Q. Z. Sheng, and L. Yao, "DFLStar: A decentralized federated learning framework with self-knowledge distillation and participant selection," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, (CIKM 2024)*, 2024.

[26] H. Huang, L. Zhang, C. Sun, R. Fang, X. Yuan, and D. Wu, "Distributed pruning towards tiny neural networks in federated learning," in *43rd IEEE International Conference on Distributed Computing Systems, (ICDCS 2023)*, 2023.

[27] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 536–550, 2022.

[28] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, (CCS 2018)*, 2018.

[29] J. Zhu, L. Wang, X. Han, A. Liu, and T. Xie, "Safety and performance, why not both? bi-objective optimized model compression against heterogeneous attacks toward AI software deployment," *IEEE Transactions on Software Engineering*, vol. 50, no. 3, pp. 376–390, 2024.

[30] P. Yin, J. Lyu, S. Zhang, S. J. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," in *7th International Conference on Learning Representations,(ICLR 2019)*, 2019.

[31] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," in *4th International Conference on Learning Representations, (ICLR 2016)*, 2016.

[32] S. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, (AAAI 2020)*, 2020.

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

[34] J. Zhu, L. Wang, and X. Han, "Safety and performance, why not both? bi-objective optimized model compression toward AI software deployment," in *37th IEEE/ACM International Conference on Automated Software Engineering, (ASE 2022)*, 2022.

[35] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, (NeurIPS 2019)*, 2019.

[36] X. He and Y. Zhang, "Quantifying and mitigating privacy risks of contrastive learning," in *ACM SIGSAC Conference on Computer and Communications Security, (CCS 2021)*, 2021.

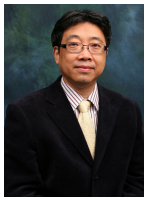
[37] J. Zhang, Z. Li, B. Li, J. Xu, S. Wu, S. Ding, and C. Wu, "Federated learning with label distribution skew via logits calibration," in *International Conference on Machine Learning, (ICML 2022)*, 2022.

Danyang Xiao received the Ph.D. degree from School of Computer Science and Engineering, Sun Yat-sen University (SYSU), China, in 2022. He is currently a postdoctoral fellow with Department of Computing, The Hong Kong Polytechnic University (PolyU), Hong Kong. Before that, he was a postdoctoral fellow with SYSU. His research interests include distributed machine learning, federated learning, and federated distillation.

Jialun Li is currently a lecturer with the School of Computer Science, Guangdong Polytechnic Normal University. He received the B.Sc. degree in computer science and technology from Central South University, Changsha, China, in 2019, and received the Ph.D. degree from School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China.

Xuan Mo received his B.S. and M.S. Degree from School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. He is currently a Ph.D. candidate in the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. His main research interests include distributed machine learning.

Weigang Wu (Member, IEEE) received the BS and MS degrees from Xi'an Jiaotong University, China, in 1998 and 2003, respectively, and the PhD degree in computer science from Hong Kong Polytechnic University, in 2007. He is currently a full professor with the School of Computer Science and Engineering, Sun Yat-sen University, China. His research interests include distributed systems and cloud computing, distributed machine learning and big data processing, blockchain and applications. He has authored or coauthored more than 100 papers published in major conferences and journals. He is on the editorial board of several international journals, including IEEE Transactions on Network and Service Management (IEEE TNSM) and Journal of Cloud Computing (JCC).



Jiannong Cao (Fellow, IEEE) is currently the Otto Poon Charitable Foundation Professor in Data Science and the Chair Professor of Distributed and Mobile Computing in the Department of Computing at The Hong Kong Polytechnic University (PolyU), Hong Kong. He is also the Dean of Graduate School, the director of Research Institute for Artificial Intelligence of Things (RIAIoT) in PolyU, the director of the Internet and Mobile Computing Lab (IMCL) . He was the founding director and now the director of PolyU's University's Research Facility in Big Data Analytics (UBDA). He served the department head from 2011 to 2017. Prof. Cao is a member of Academia Europaea, a fellow of the Hong Kong Academy of Engineering Science, a fellow of IEEE, a fellow of China Computer Federation (CCF) and an ACM distinguished member.