

PRISAM: Efficient Personalization via BN Masks in Heterogeneous Decentralized Federated Learning

Shan Chang*, Xianbo Wang*, Hao Yu*, Denghui Li*, Guanghao Liang*, Hongzi Zhu[†] and Bo Li[§]

*Donghua University [†]Shanghai Jiao Tong University

[§]Hong Kong University of Science and Technology

changshan@dhu.edu.cn, {2212508, 2242835, 2222731, 2232843}@mail.dhu.edu.cn

hongzi@cs.sjtu.edu.cn, bli@cse.ust.hk

Abstract—Decentralized Federated Learning (DFL) removes the central server in traditional Centralized FL, eliminating performance and security bottlenecks while improving scalability for large-scale cross-device federated scenarios. In these scenarios, devices are heterogeneous in computation, storage, and data distribution, requiring personalized models. A common strategy for personalized DFL is for each device to collaborate with others having similar data distributions to train a federated model and then perform local pruning. However, in the absence of a central server, along with privacy concerns and the limited inference capabilities of low-end devices, challenges emerge in terms of communication, computation, and model convergence. This paper presents an efficient personalized DFL method, named PRISAM, based on BN (Batch Normalization) masks. Each device adaptively adjusts its BN mask, enabling effective structured pruning with minimal performance loss. By exchanging BN masks, communication overhead for similarity comparison is reduced by a factor of 100,000, comparing to exchanging an entire model, while computational costs are also significantly lowered. Extensive experiments show that PRISAM significantly improves personalized model performance on three datasets across two models, outperforming state-of-the-art methods, while greatly reducing computational and communication overhead.

Index Terms—decentralized federated learning, BN mask, structural pruning, personalization, heterogeneity

I. INTRODUCTION

Federated Learning (FL) [1]–[5] enables collaborative model training across devices without sharing private data. Traditional Centralized FL (CFL) relies on a central server for model aggregation, which creates scalability bottlenecks and security risks. Decentralized FL (DFL) [6] [7] removes the central server, allowing devices to collaborate directly in a peer-to-peer (P2P) manner. The decentralized nature of DFL offers several advantages, including enhanced privacy, scalability, and resilience to single points of failure.

Heterogeneity is prevalent in practical DFL scenarios. First, data distributions across devices are often non-IID (non-independent and identically distributed), which complicates the creation of a universally effective global model. For instance, in a text prediction task, users may display distinct typing patterns, language preferences, and frequently used vocabulary, leading to diverse and uneven data distributions. Additionally, devices vary significantly in terms of computational power, memory, and communication bandwidth, resulting in uneven training capabilities. For example, low-power

IoT devices may have limited processing resources and memory, while more powerful smartphones or edge servers offer considerably greater computational capacity. Consequently, it is crucial to train personalized models tailored to each device, ensuring alignment with its data distribution, computational, and storage constraints, though this remains a challenging task.

Personalized training in federated learning (FL) requires identifying devices with similar data distributions for collaboration, but the lack of a central coordinator and privacy concerns complicate device discovery. In large-scale decentralized FL systems, this process adds significant overhead in communication, computation and storage. Additionally, heterogeneous model architectures across devices create challenges in federated aggregation, requiring alignment of disparate models while preserving system efficiency.

In existing personalized DFL approaches, devices compare their local models to infer data distribution similarities [8]. However, directly exchanging models leads to significant communication overhead and exposes vulnerabilities to membership inference attacks. Additionally, as model parameter dimensions increase, identifying effective similarity metrics becomes more complex. Displ [8] randomly selects a subset of devices for aggregation, so as to mitigate communication overhead of similar device discovery. However, this strategy limits personalized model performance, as it fails to capture the unique characteristics of each device's data distribution. In model light-weighting, pruning the global model is widely adopted, as it allows local models to align with the global architecture, simplifying aggregation. Pruning can be structured or unstructured, depending on the granularity of the pruned weights. Most methods use unstructured pruning [8], which maximizes sparsity. However, despite sparse weights, dense intermediate activations still require specialized hardware or software for efficient inference.

In this paper, we present PRISAM, an efficient DFL protocol specifically designed to address the challenges posed by data and device heterogeneity in decentralized environments. First, we propose a structured pruning method using BN (Batch Normalization) [9] masks, represented as binary sequences. Channels with a mask value of *zero* are considered pruned. Devices dynamically adjust the sparsity of BN masks according to their computational and storage constraints,

achieving adaptive pruning. BN layer pruning leverages the importance of channels, as reflected by their corresponding scaling factors, ensuring that the most relevant features are retained and minimizing performance degradation compared to other pruning strategies. Meanwhile, under the guidance of BN masks, heterogeneous models can efficiently perform federated aggregation.

Second, PRISAM significantly reduces communication costs by exchanging BN masks instead of full models when identifying devices with similar data distributions. The intuition is that local models trained on statistically similar data tend to exhibit similar BN layer characteristics. This approach leads to a dramatic decrease in communication volume. For instance, in VGG11, the full model size is 35.38 MB, whereas the BN mask is just 344 bytes—resulting in over a 100,000-fold reduction in communication cost. Additionally, exchanging BN masks minimizes privacy risks associated with membership inference attacks.

Third, PRISAM extensively reduces the computational overhead of device similarity comparisons by simplifying model-level comparisons into BN mask comparisons. Initially, it examines BN masks from different devices and observes that the vast majority of mask values are globally consistent (i.e., the mask values associated with a particular channel are consistently 1 or 0 across all models.), likely due to shared structural characteristics. Then, these globally consistent mask values are eliminated before conducting comparisons, and the remaining BN mask sequences are clustered using an unsupervised approach, enabling efficient and accurate identification of devices with similar data distributions.

We evaluate PRISAM on three datasets—CIFAR-10, CIFAR-100, and SVHN—using two models (VGG11 and ResNet) and three different Non-IID data partitions (two Dirichlet distributions with different parameters and a pathological distribution). The experimental results show that PRISAM outperforms three baseline methods in terms of model convergence, accuracy, communication efficiency, and computational overhead. Furthermore, we investigate the impact of pruning rate, the number of heterogeneous distributions, device dynamics, and device heterogeneity on PRISAM's performance. The experimental results demonstrate the robustness of PRISAM under various impact factors.

II. RELATED WORK

This section reviews the literature studies including personalized FL, decentralized FL and model pruning.

A. Personalized FL

Yang *et al.* [10] present a Group-based Federated Meta-Learning framework for obtaining the personalized models via meta-learning. The grouping mechanism guarantees that the devices in each group have similar data distribution to achieve personalization. Jin *et al.* [11] design a novel personalized FL scheme by the self-knowledge distillation, where devices can derive the knowledge of previous personalized models to current local models. Lin *et al.* [12] propose an ensemble

distillation scheme for model fusion. The central classifier is trained based on these unlabeled data on the outputs of the models from the devices, which mitigates privacy risk and cost. Li *et al.* [13] consider that each device can design its models independently, where the transfer learning and knowledge distillation are leveraged for promoting FL when each agent owns their private data and the designed models. Zhang *et al.* [14] present a training scheme using personalized models for different devices, in which each device can maintain a personalized soft prediction to guide local training, while the aggregation procedure is performed as a personalized group knowledge transfer training algorithm. Briggs *et al.* [15] propose a hierarchical clustering step for FL, with the objective of separating devices into clusters based on the similarity of their local updates, in which the clusters are allowed to train their models in parallel. Long *et al.* [16] design a multi-center aggregation framework in personalized decision-making system to cluster devices based on the model parameters, where the multi-center federated loss is formulated for user clustering in FL. Ghosh *et al.* [17] propose a clustered FL framework, and the iterative federated clustering algorithm (IFCA) is proposed for estimating the cluster identities of the users and optimizing model parameters. To address the data imbalanced issue, Chou *et al.* [18] propose a global regularized personalization (GRP-FED), and an adversarial discriminator is adopted for regularizing between the learned global-local features to prevent the local model from overfitting. Shen *et al.* [19] introduce a cyclic distillation-guided channel decoupling scheme in FL, with the goal of achieving more accuracy and consistent regularization between local and global model representations.

B. Decentralized FL

Dai *et al.* [8] propose Dis-PFL, a personalized FL framework under decentralized communication protocols, which solves the problem of data and device heterogeneity by employing a masking technique that enables devices to obtain sparse personalized networks. Ma *et al.* [20] utilize the cosine similarity of the gradient to guide neighbor selection based on adaptive model pruning and communicate with the selected neighbors through a peer-to-peer (P2P) approach, thus reducing the communication overhead of decentralized federated learning. Beltrán *et al.* [6] study the fundamentals of decentralized FL from the perspective of communication mechanism and security, and the optimized decentralized FL fundamentals are discussed. Sui *et al.* [21] present FederiCo, a decentralized and personalized FL framework, in which a communication-efficient protocol is designed for fully-decentralized learning. Simulations have validated the performance in comparison with several benchmark datasets. Chen *et al.* [22] design a privacy-preserving decentralized FL architecture, which ensures the confidentiality of model gradients against internal and external adversaries through the dropout-tolerated aggregation scheme. Kalra *et al.* [23] investigate ProxyFL, a communication-efficient framework for decentralized FL, which eliminates the limitation of canonical FL by allowing

model heterogeneity and achieves privacy preservation via differential privacy. Xie *et al.* [24] study a decentralized FL scheme and propose an asynchronous parameter-sharing algorithm, in which a joint node scheduling and bandwidth allocation is formulated to minimize the system delay. Tang *et al.* [25] propose GossipFL, a decentralized FL framework, in which the gossip matrix generation algorithm is introduced to improve the resource utilization while guaranteeing the convergence property. Li *et al.* [26] present a decentralized FL based on the mutual knowledge transfer, in which local devices fuse models by transferring the learned knowledge to each other while guaranteeing the learning convergence.

C. Model Pruning

Jiang *et al.* [27] propose a pruning mechanism based on complement sparsification (CS), where the CS creates a global sparse model consisting of the weights of data distribution of all devices, and the devices then create local sparse models with the weights pruned from the global model. Huang *et al.* [28] present a distributed pruning FL framework, in which an adaptive batch normalization selection module and lightweight progressive pruning module are introduced to prune the models. Liu *et al.* [29] propose a novel learning scheme by utilizing convolutional neural networks (CNNs), which can reduce the model size and decrease the number of computing operations, while ensuring learning accuracy. By pruning redundant connections via the three-step method, Han *et al.* [30] present an efficient scheme for training using a neural network with the objective of reducing overheads of the storage and computation. Li *et al.* [31] focus on improving the performance of well-trained CNNs through pruning filters, where an acceleration method is proposed for CNNs to reduce the effect on the model accuracy. Changpinyo *et al.* [32] consider that each output convolution channel can be connected to a small random fraction of the input channels by a sparse connection matrix, which improves the computation efficiency since the channel spatial convolution remains unchanged. Wen *et al.* [33] introduce a structured sparsity learning (SSL) approach for regularizing the structures of Deep Neural Networks (DNN), which promotes the well-regularized big models with improved accuracy and computation efficiency. Li *et al.* [34] propose Hermes, a communication and inference-efficient FL scheme, which performs averaging on these intersections of subnetworks while keeping the remaining non-intersected elements unchanged and protecting the privacy of devices. Jiang *et al.* [35] introduce PruneFL, an adaptive and distributed parameter pruning scheme, in which the model is adjusted adaptively to reduce both communication and computation latency during FL, while guaranteeing the model accuracy.

III. SYSTEM MODEL AND PROBLEM DEFINITION

A. System Model

Federated devices:

We consider a DFL system with \mathcal{N} devices, denoted by $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_{\mathcal{N}}\}$, that operates using P2P communication

model. In this system, there is no central server acting as a coordinator, and all devices are treated as equal peers. However, the devices are heterogeneous in terms of their computational and storage capabilities, as well as in the statistical distribution of their local data.

Data Heterogeneity: each device \mathcal{D}_i ($i \in \{1, 2, \dots, \mathcal{N}\}$) possesses a local private dataset \mathbb{D}_i , which is assumed to be Non-IID (Non-Independent and Identically Distributed) across devices. Moreover, we consider that each device does not have sufficient local data to train an accurate model solely by itself, thus it should collaborate with others.

Device Heterogeneity: refers to the differences in the capabilities and characteristics of the devices participating in the DFL system. These differences can include variations in computational power, storage capacity, network bandwidth, and energy resources.

the system employs a peer-to-peer communication approach. Model sharing among devices can use hybrid protocols, e.g., pointing [36], gossip [37], or broadcast [38]. At each communication round, devices may obey different protocols to transmit models to one or more other devices.

During the initialization phase, each device initializes the weights of the same model structure according to the same rule, resulting in an initialized model W_0 . Building upon this, in each communication round, the initialized model undergoes e rounds of training using the local private data at each device. At the beginning of each communication round during training, each device follows the agreed-upon model synchronization protocol. Through peer-to-peer communication, devices either receive or send the model and participate in model aggregation. The federated learning training concludes upon reaching the pre-agreed R rounds or when the global device loss reaches a minimum loss target value l_{min} .

B. Problem Definition

Due to the heterogeneity on data distribution, resources, and tasks of devices, they no longer share the same model structure and weights. Instead, each possesses its personalized model.

Definition Heterogeneous Decentralized Federated Learning. For each device \mathcal{D}_i , its local dataset $\mathbb{D}_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{n_i}$ is drawn from a distribution $p_i(x, y)$, where $x_{i,j}$ and $y_{i,j}$ are the training samples and the corresponding labels, respectively, and n_i is the size of \mathbb{D}_i . For $i \neq k$, $p_i(x, y) \neq p_k(x, y)$.

During federated training, \mathcal{D}_i selects devices $\mathcal{N}_i \subset \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_{\mathcal{N}}\}$, whose data distributions $p_k(x, y)$ are similar to $p_i(x, y)$. \mathcal{D}_i learns a personalized model \mathcal{W}_i based on \mathcal{D}_i and models \mathcal{W}_k from devices in \mathcal{N}_i . The local training objective for \mathcal{D}_i is:

$$\mathcal{L}_i(\mathcal{W}_i) = \frac{1}{n_i} \times \sum_{j=1}^{n_i} \mathcal{L}(x_{i,j}, y_{i,j}, \mathcal{W}_i),$$

the objective function of personalized federated training is,

$$\min_{\mathcal{W}_i} \mathbb{L}(\mathcal{W}_i) = \frac{1}{\mathcal{N}_i} \times \sum_{i=1}^{\mathcal{N}_i} \mathcal{L}_i(\mathcal{W}_i),$$

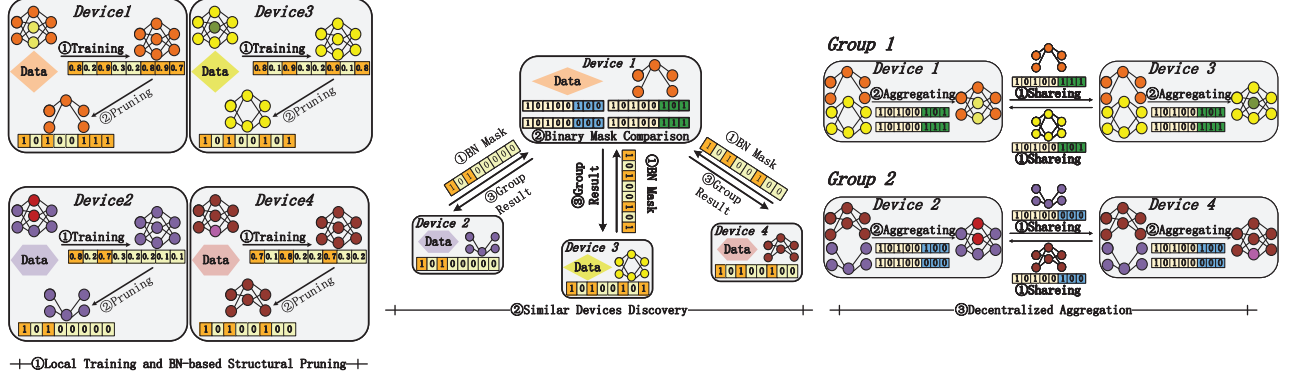


Fig. 1: PRISAM Framework

On the other side, the communication cost between devices becomes a crucial factor that cannot be ignored. Therefore, while considering the objective loss function, we also aim to optimize the communication cost:

$$\min_{\mathcal{R}} \mathbb{T} = \mathcal{R} \times (up(\mathcal{W}) + down(\mathcal{W}))$$

Where \mathcal{R} represents the pre-determined number of model updating iterations. up and $down$ represent the amount of data uploaded and downloaded by devices during one communication round, respectively.

IV. PRISAM DESIGN

A. Overview

PRISAM consists of four phases (see Figure 1), i.e., Initialization, Local Training and BN-based Structural Pruning, Similar Devices Discovery and Decentralized Aggregation.

Initialization: all federated devices collaboratively negotiate the full structure of a model \mathcal{W} to be trained, and key parameters, e.g., the number of local training in each iteration.

Local Training and BN-based Structural Pruning: each device \mathcal{D}_i uses its private dataset \mathbb{D}_i to initialize its local model as $\mathcal{W}_i^{(0)}$. Next, it performs structural pruning on $\mathcal{W}_i^{(0)}$ according to its pruning ratio ρ_i , obtaining a simplified model $\widetilde{\mathcal{W}}_i^{(0)}$ and the corresponding structural BN mask $\mathcal{M}_i^{(0)}$.

Similar Devices Discovery: in iteration t , devices exchange their BN masks $\mathcal{M}_i^{(t)}$. Devices with similar data distributions are grouped by applying unsupervised learning to these masks. Before that, useless dimensions will be removed to obtain a compact mask $\overline{\mathcal{M}}_i^{(t)}$, which can improve grouping accuracy, as well as reduce communication and computational cost.

Decentralized Aggregation: each device \mathcal{D}_i receives the pruned models $\widetilde{\mathcal{W}}_j^{(t)}$ of the devices within the same group, as well as the corresponding masks $\mathcal{M}_j^{(t)}$. After aligning the structures of pruned models with their BN masks, heterogeneous models $\widetilde{\mathcal{W}}_j^{(t)}$ are aggregated to obtain the group model $\mathcal{W}_{\mathbb{G}_k}^{(t)}$, where \mathbb{G}_k denotes the k -th group of devices. Then, $\mathcal{W}_{\mathbb{G}_k}^{(t)}$ is considered as the initial local model of the next iteration.

The second to fourth phases are executed iteratively until the predetermined number of iterations is completed, or the model converges (Algorithm 1).

Algorithm 1 PRISAM

- 1: **Input:** Total number of devices \mathcal{N} ; Prune ratio ρ_i for each device; Number of iterations T
 - 2: **Local Training and BN-based Structural Pruning:** Each device \mathcal{D}_i uses its local private dataset \mathbb{D}_i to initialize its local model as $\mathcal{W}_i^{(0)}$; Then it uses its pruning ratio ρ_i to obtain the simplified model $\widetilde{\mathcal{W}}_i^{(0)}$ and the corresponding structural BN mask $\mathcal{M}_i^{(0)}$
 - 3: **Output:** Personalized local model $\widetilde{\mathcal{W}}_i^{(T)}$
 - 4: **for** $t = 0, 1, \dots, T - 1$ **do**
 - 5: Share mask $\mathcal{M}_i^{(t)}$ with other devices
 - 6: Compact mask $\overline{\mathcal{M}}_i^{(t)} = \text{Compact}(\mathcal{M}_i^{(t)})$ # Remove useless dimensions
 - 7: Group devices to obtain \mathbb{G}_k based on similar data distributions using $\overline{\mathcal{M}}_i^{(t)}$
 - 8: Download_To_Get $\{\widetilde{\mathcal{W}}_j^{(t)}, \mathcal{M}_j^{(t)}\}_{j \in \mathbb{G}_k, j \neq i}$
 - 9: Align the structures of pruned models with their corresponding masks
 - 10: $\mathcal{W}_{\mathbb{G}_k}^{(t)} = \text{Aggregate}(\{\widetilde{\mathcal{W}}_j^{(t)}\}_{j \in \mathbb{G}_k})$
 - 11: $\mathcal{W}_i^{(t+1)} = \text{Train}(\mathcal{W}_{\mathbb{G}_k}^{(t)}, \mathbb{D}_i)$ # Train with local data \mathbb{D}_i
 - 12: $\widetilde{\mathcal{W}}_i^{(t+1)}, \mathcal{M}_i^{(t+1)} = \text{Prune}(\mathcal{W}_i^{(t+1)}, \rho_i)$ # Obtain pruned model and mask for next iteration
 - 13: **end for**
-

BN-based Structural Pruning of Local Models: each device \mathcal{D}_i uses its private data to train its current group model $\mathcal{W}_{\mathbb{G}_k}^{(t)}$, leading to a new local model $\mathcal{W}_i^{(t+1)}$. Next, it performs structural pruning on $\mathcal{W}_i^{(t+1)}$ according to ρ , obtaining a simplified model $\widetilde{\mathcal{W}}_i^{(t+1)}$ and the corresponding structural binary mask $\mathcal{M}_i^{(t+1)}$.

The second to fourth phases are executed iteratively until the predetermined number of global iterations is completed, or the model converges.

B. BN-based Structural Pruning

The BN layer can usually be added before the activation function of the deep neural network, and the activation values are normalized after the convolutional layer. It mainly limits the activation value of each layer to a reasonable range by normalising the activation value of each layer, which can effectively prevent gradient vanishing and gradient explosion and accelerate the training of deep network models.

When the intermediate activation values of the neural network model pass through a BN layer, the mean and variance are calculated for all intermediate activation values for a small batch of samples \mathcal{B} : $\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^m x_i$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$$

where m is the number of samples in batch \mathcal{B} , x_i represents the intermediate activation value output by the i -th sample after passing through the previous convolutional layer, and the intermediate activation x_i of each sample is normalized by the mean $\mu_{\mathcal{B}}$ and variance $\sigma_{\mathcal{B}}^2$ of the derived small batch samples:

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

After normalizing the activation values, the intermediate activation values are recovered by means of the learnable reconstruction parameters γ and β in order to recover the distribution of features to be learned by the original network:

$$y_i = \gamma \hat{x}_i + \beta$$

where y_i is the output activation value of the current BN layer, which is used as input for the next layer. For each BN layer the output, y_i can be transformed into the input x_i through a linear transformation of the parameter γ . Liu *et al.* describe γ as a scaling factor for intermediate activation values, and the model structure is optimized by determining the importance of the intermediate activation values corresponding to them by the value of γ , and pruning the channels corresponding to relatively unimportant activation values.

Since each intermediate activation of the inputs arriving at the BN layer is multiplied by this scaling factor, a scaling factor that is too small results in its corresponding BN layer output activation value also being small, and has less impact on the connections that follow it, and thus can be considered relatively unimportant. When pruning, the scaling factors of each layer are sorted by absolute value, and the scaling factor to be pruned is selected according to the pre-set pruning rate, and the structure of the previous and subsequent layers corresponding to this scaling factor will also be removed from the model structure.

As shown in Figure 2, the filter output activation values are normalized and then need to be linearly transformed by the scaling factor γ . When a pruning rate of 25% is chosen, then 25% of the channels are cropped in each layer. After sorting the scaling factors for the i -th BN layer, the scaling factor with a value of 0.005 has the smallest value and

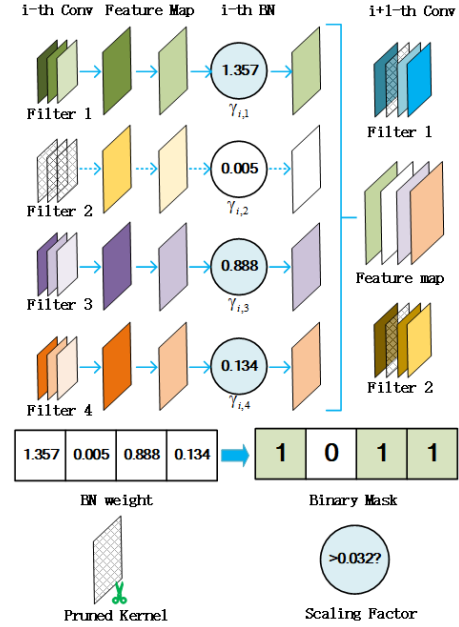


Fig. 2: an example of BN-based pruning and binary BN mask constructing.

is selected for deletion, and its corresponding filter in the previous layer and convolution kernel in the subsequent layer will be removed from the network model structure. In order to represent the pruned model structure in a more efficient way, the weights through the BN layers can be assigned a mask value corresponding to each position. When a scaling factor is selected for deletion in the current layer due to its relatively small value, the mask value of its corresponding position is set as *zero*, indicating that the current position is deleted, otherwise the mask of its corresponding position will be set as *one*, indicating that the current position is retained.

After pruning the model, the model structure is modified to obtain a simpler model with a binary sequence corresponding to the model structure before and after pruning. Experiment shows that for a VGG11 model with BN layers, the length of the corresponding mask sequence \mathcal{M} is only around 2700 bits. Since the pruned model thus obtained is a modification of the model directly at the structural level, rather than a sparsification of the model weights, it is possible to achieve speedups in model inference or training without special hardware support.

C. Discovering Similar Devices via Binary Mask Comparison

Given a pruned model \mathcal{W}_i , the device p_i holds a mask of binary sequence \mathcal{M}_i , representing which part of the model is pruned. The structure of the model can to some extent characterize the distribution of the corresponding training dataset. Therefore, this enlightens us to assist devices in globally searching for neighbors with similar data distribution through comparison of binary masks.

Furthermore, intuitively, a large fraction of the channels preserved by different devices are the same. (Subsequent

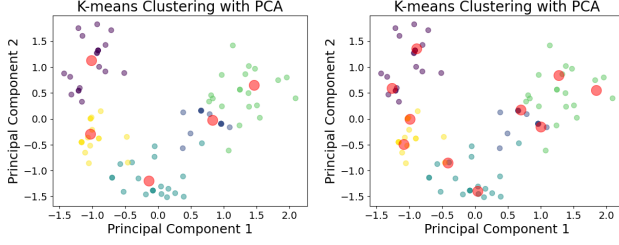


Fig. 3: Visualization of BN mask clustering

Algorithm 2 Grouping Devices

```

1: Input: Total number of devices  $\mathcal{N}$ ; Pruning ratio  $\rho$ 
2: Initialization: Coherently initialize each device's model  $\mathcal{W}_{k,0}$ ; Train the model with local privacy data and prune it to obtain the mask  $\mathcal{M}_k$ 
3: Output: A collection of devices in the same group  $G_k$ 
4: for node  $k$  in parallel do
5:   Receive neighbors' masks  $\mathcal{M}_k$ 
6:   for  $d = 0$  to  $D$  do
7:      $sum_d = \sum_{k=1}^{\mathcal{N}} \mathcal{M}_{k,d}$ 
8:   end for
9:   if  $sum_d == K$  or  $sum_d == 0$  then
10:    All masks  $\mathcal{M}$  remove the  $d$  dimension
11:   end if
12: end for
13: for node  $k$  in parallel do
14:    $G_k = \text{Unsupervised\_Learning}(\mathcal{M})$ 
15: end for

```

experiments demonstrate it. For example, in the mask sequences of the VGG11 network belonging to 100 devices with non-IID setting (CIFAR10, Dirichlet distribution with $\alpha = 0.2$), only 13 dimensions are different, which also implies that all devices' pruned models retained the vast majority of the consistent channels, and the structural differences in pruning are only reflected in a very small fraction of these channels.) Therefore, before mask comparison, PRISAM first compresses the masks. In detail, if it is found that, in a certain dimension, the pruned models of all devices have the same mask value (one or zero represents all devices' models retain the corresponding scaling factor, or remove their corresponding channels), then this dimension will be removed. In other words, the common parts of the masks are removed. Given a set of original masks, for an original mask \mathcal{M} , after dimension removal, a compact mask sequence $\bar{\mathcal{M}}$ can be obtained for comparison. Then, unsupervised learning (e.g., KMEANS, DBSCAN) algorithm can be applied on $\bar{\mathcal{M}}_i$, $i \in [\mathcal{N}] = 1, 2, \dots, \mathcal{N}$ (see Algorithm 2) to group all devices.

Mask compression can not only effectively help devices find others with similar data distribution for collaboration, but also further reduce communication and computational overhead, thereby achieving better scalability. In practice, masks are not necessarily sent to all devices for compression. Instead,

devices can negotiate to select one device to collect all masks and conduct both mask compression and clustering in each iteration.

Fig. 3 visualizes the clustering result (the compact masks are grouped into *five* and *ten* groups in (a) and (b), respectively.) of compact masks of 100 pruning models (from devices with CIFAR-10, $\alpha = 2$) by mapping the sequences into a two-dimensional space using PCA. We group the masks simply utilizing K-means and obtain the centroids for each group. In the figure, compact masks are represented by small circles with different colors, while the red bigger circles denote the centroids of groups.

D. Decentralized Learning of Personalized Models

For each device p_i , the models it receives come from local models trained by group members with similar data distributions, and thus aggregation with these models can effectively optimize the personalized models of p_i (structure and weights). However, since the devices hold personalized pruned model, which implies their structures differ from each other. When the models are aggregated, the structures of the device models will very likely be changed. However, if only the pruned models are shared, will lead to the inability to complete the model aggregation. Therefore, it is necessary to simultaneously transmit the masks \mathcal{M} carrying model structure information to indicate the position of the pruned model structure in the original model, and the receiver performs positional alignment of the received model structure based on \mathcal{M} . In other words, all the weights of the pruned model are mapped to the corresponding channels of the complete model through the mask for model structure recovery.

Next, weight averaging is performed on the recovered model, after which the local model is re-pruned according to the weights of the BN layer after the aggregation for the purpose of optimising the model structure. In other words, in personalized model aggregation, due to the differences in the model structure of each device, the structure that was pruned out in the previous iteration may be restored (corresponding to a larger value of the scaling factor of the BN layer) by regaining its relative importance after the aggregation in the current round, and on the contrary, the structure that was retained in the previous round may be pruned out as a result of model aggregation.

As depicted in Fig. 4, for the weights of the three models that need to be aggregated, if a weight is retained by all models, the aggregated model weight, following the method of model averaging, equals the average of the corresponding weights of the three models. When a weight is retained by some of the models, the weight of the aggregated model corresponds to the average of the retained weights. If a weight is only retained by one of the models, the aggregated model weight equals the weight retained by that model. If a portion of the weight is deleted by all models, then the final aggregated model will also lack that portion of the weight. Regarding the aggregated model, devices need to select a subset of the

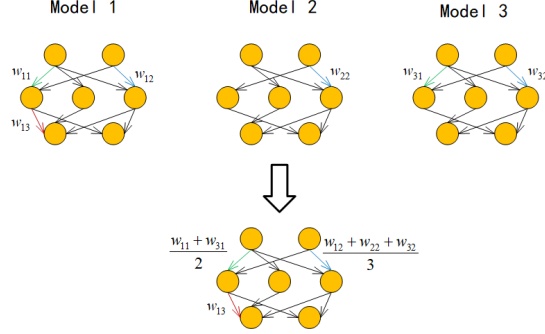


Fig. 4: An illustration of the aggregation of pruned models

aggregated model weights based on their own structure as updates to the local model.

V. EXPERIMENT

A. Methodology

Our experimental environment is based on Python 3.9 as well as Pytorch 2.1, and we conduct a series of experiments on a server (Intel Xeon Silver 4316x2, RTX3090 x4) for testing. During the model training process, the neural network model is updated by the stochastic gradient descent (SGD) method, and the learning rate is set to 0.1. To simulate the real data distribution in the actual deployment of Federated Learning, a total of 200 devices are set up globally and divided into groups. After initialization, each device undergoes 3 rounds of local training and is subsequently clustered based on the BN masks generated during the local training. In each iteration, 100 devices are selected for federated learning.

Datasets. We test PRISMA on CIFAR-10, CIFAR-100, and SVHN datasets, considering two types of data partitioning that are commonly encountered in real-world federated learning deployments.

- 1) *Dirichlet Cluster Partition*: the devices are divided into k groups, with the data distribution between the groups following a Dirichlet distribution with parameters $\alpha = 0.2$ and $\alpha = 1$, respectively. The data distribution across devices within each group is IID.
- 2) *Pathological Cluster Partition*: each group contains a subset of classes from all available classes, and the data distribution across devices within the group is IID. For CIFAR-10 and SVHN, each group consists of two classes of samples, while for CIFAR-100, each group contains 20 classes. The pathological data partition is referred to as n -cls in the results.

Models. We chose two models with different sizes.

- 1) *VGG11*: consists of a total of 11 layers, including convolutional layers, pooling layers, and fully connected layers. Out of these 11 layers, 8 are convolutional layers (3x3 convolution), and 3 fully connected layers connected by a maximal pooling layer and a ReLU activation function. To apply structured pruning methods, BN layers are added after the convolutional layers. We

use CIFAR-10 and CIFAR-100, respectively, to train VGG11.

- 2) *ResNet-20*: a shallow variant designed for small datasets. The network consists of an initial convolutional layer, 3 residual modules (each containing 3 Bottleneck blocks), and a fully connected classification layer, with a total depth of 20 layers. It extracts features using residual connections and global average pooling. BN layers are added after the initial convolutional layer, as well as each convolutional layer in Bottleneck blocks. It well-suited for classification tasks with small-sized images. We use SVHN to train it.

Baselines. We compare PRISMA to baseline methods as follows:

- 1) *Local Training without Pruning*: each device is trained only on its local data for 100 rounds and evaluated directly without any pruning.
- 2) *FedAvg with random device selection*: in each iteration, each device randomly selects a subset of devices to collaborate on federated training, without pruning. Local model updates are aggregated using FedAvg.
- 3) *Dis-PFL*: A state-of-the-art decentralized personalized federated learning method. In each iteration, each device randomly selects a subset of devices to collaborate with. The device first downloads the current local models from the selected devices and aggregates them into a federated model. Afterward, the device personalizes the federated model by applying structural pruning on it.

B. Overall Performance Comparison

In this experiment, we randomly selected 100 devices and divided them into 5 groups to evaluate the performance of personalized models under three different data distributions, with a pruning rate of 0.5. The results, shown in Figure 5, indicate that PRISMA consistently outperforms other methods across all experimental settings. It is important to note that both Local and FedAvg represent the accuracy of the unpruned baseline models. Even after BN pruning, the personalized models trained with PRISMA still achieved better performance, comparing with Local and FedAvg. When compared to Dis-PFL, under the same pruning rate, PRISMA demonstrates clear advantages, primarily due to: 1) BN pruning leading to smaller performance loss, and 2) using devices with similar data distributions rather than randomly selected devices. Additionally, PRISMA performs best in pathological data distributions, showing a significant improvement over methods that aggregate models from randomly selected devices. For example, in the CIFAR-10 task with VGG11, under pathological distributions, the average accuracy of the personalized models from 100 devices trained with PRISMA is 92%, while the accuracy of Dis-PFL was only 59%. This is because, as the data distribution becomes more imbalanced, random device selection leads to a greater mismatch between the overall training data distribution and the actual task distribution.

Comparison of Different Distributions in Different Datasets

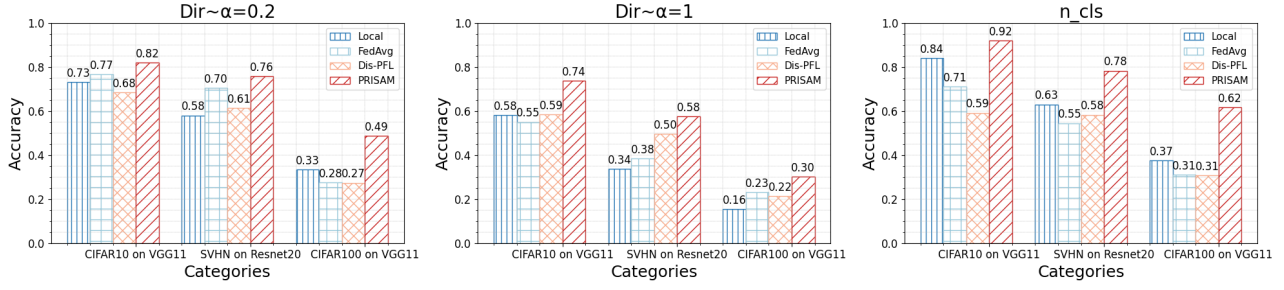


Fig. 5: Model performance comparison

TABLE I: CIFAR10 on VGG, $\alpha = 0.2$ and Pathological

Methods		Acc		Com	Flops
		Dirichlet $\alpha=0.2$	Pathological		
PRISMA	$\rho=0.3$	81.23	89.03	170	75
	$\rho=0.5$	80.28	90.88	87	38
	$\rho=0.7$	80.57	90.57	31	14
Local		73.57	84.35	0	153
FedAvg		68.48	50.63	352	153
Dis-PLF $\rho=0.5$		76.53	50.38	352	153

TABLE II: SVHN on ResNet, $\alpha = 0.2$ and Pathological

Methods		Acc		Com	Flops
		Dirichlet $\alpha=0.2$	Pathological		
PRISMA	$\rho=0.3$	74.98	78.41	11	7.62
	$\rho=0.5$	75.88	75.90	9	16.32
	$\rho=0.7$	75.97	57.54	3	3.23
Local		58.11	63.11	0	35.36
FedAvg		70.50	54.56	17	35.36
Dis-PLF $\rho=0.5$		61.49	58.39	17	35.36

C. Communication and Computing Resource Consumption

In Tables I and II, we further compare the proposed method with other approaches in terms of communication and computational overhead. In this experiment, 100 randomly selected devices are evenly divided into 5 groups. For the local training method, devices only use their local private datasets to train the model without communicating with other devices, resulting in no communication overhead. For the FedAvg method without pruning, each device aggregates models from 19 other randomly selected devices in each round, receiving the complete model weights from each device, which results in significant communication overhead. In the case of Dis-PFL, since it employs weight-level pruning, the complete weights still need to be transmitted during the communication process, which presents a significant challenge for devices with limited communication bandwidth. For our proposed method, with a pruning rate of 0.5, approximately 3/4 of the model weights are pruned. As a result, the communication volume is greatly reduced, leading to improved model accuracy.

By pruning the model, we obtain a more compact version,

significantly reducing the hardware resources required for deploying the trained model in subsequent use. For example, as shown in Table I, setting the pruning rate to 0.7 in our method results in a compact model that requires less than 1/10 of the floating-point operations needed for inference compared to the full model. In contrast, for the pruning method used in Dis-PFL, without specialized hardware and software library support, it's not possible to optimize computations for sparse matrices. As a result, the inference speed remains the same as that of the full model, which has not undergone pruning. For other methods, since no model optimization is applied, the final trained models consume more hardware resources during inference.

D. The Impact of Pruning Rate on Performance

Model pruning can help reduce the risk of overfitting during training, and in some cases, it may even improve performance compared to using the full model. However, the pruning rate has a critical impact on the pruned model's performance. Excessive pruning can result in the loss of important information, leading to reduced performance. In our approach, the pruning structure of the model plays a key role in helping devices identify suitable collaborators. Models with light pruning may not effectively capture the characteristics of local data distributions, making it harder to find suitable devices for collaboration and reducing the benefits of federated learning from local training. Therefore, it is important to adjust the pruning rate to strike a balance between local computational resource usage and model performance.

We conduct experiments on CIFAR-10 using VGG and SVHN using ResNet, and compare PRISMA with Dis-PFL. We also randomly select 100 users and divide them into 5 groups, and use two data distributions: the Dirichlet distribution with parameters $\alpha = 0.2$ and a Pathological distribution with $n_cls=2$. The model's pruning rate is varied at 0.3, 0.5, and 0.7, meaning that 30%, 50%, and 70% of the BN mask values are set as *zero*, respectively, then the corresponding channels are pruned. As shown in Fig. 7, across both data distributions and tasks, setting the pruning rate to 0.5 consistently yields the best model performance. This is likely because a pruning rate of 0.5 strikes an optimal balance between preserving data distribution characteristics

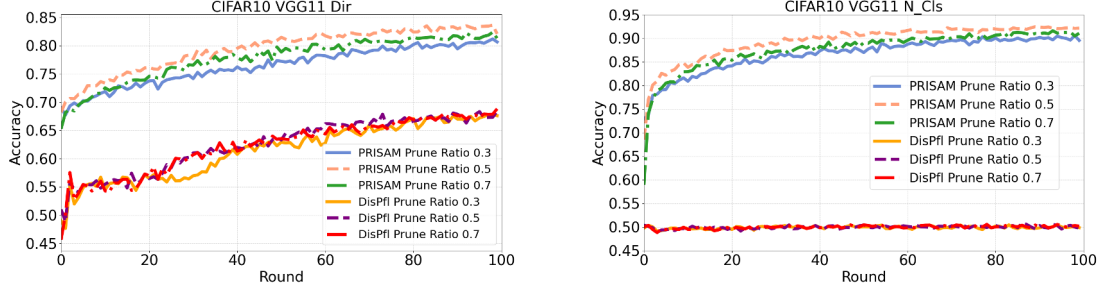


Fig. 6: Model accuracy change with training iterations under different ρ (CIFAR-10 on VGG11).

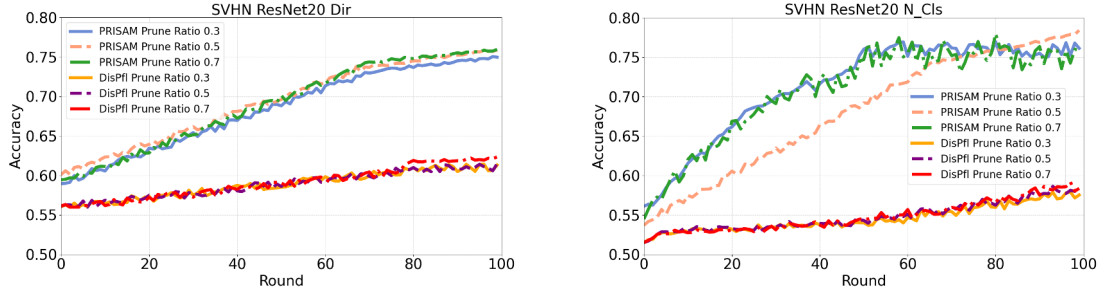


Fig. 7: Model accuracy change with training iterations under different ρ (SVHN on ResNet).

and maintaining the model's expressive power. When the pruning rate is set to 0.3, only 30% of the BN layer channels, corresponding to the positions where the BN mask is set to *zero*, are pruned, which helps retain the model's expressive capacity. However, this insufficient pruning representation of local data distributions makes it challenging to effectively identify collaborating devices from a global perspective. Conversely, when the pruning rate is set to 0.7, 70% of the BN layer channels are pruned. While this pruning reflects the local data distribution more accurately, excessive pruning reduces the model's expressiveness, leading to poor convergence and ultimately worse performance.

From another perspective, PRISMA significantly outperforms Dis-PFL in terms of both convergence speed and the final model accuracy across both tasks and data distributions. For example, in the task of SVHN on ResNet20 with a pathological data distribution, PRISMA achieves an average model accuracy of 76%, while Dis-PFL only reaches 57%. Moreover, PRISMA converges in around 60-th rounds, which means the number of training rounds can be reduced by 40%.

E. Impact of Group Number on Performance

We conduct experiments on CIFAR-10 using VGG and SVHN using ResNet. For PRISAM, we randomly select 72 devices and divide them into 3, 6, 9, and 12 groups for the experiments. In the comparison experiment, each device randomly selects 10 BN pruned models from other devices for aggregation. The inter-group data distribution follows a

Dirichlet distribution with parameters $\alpha = 0.2$. The pruning rate is set to 0.5. In this experiment.

The experimental results in Fig. 8 indicate that the number of groups k has a significant impact on the model performance. In general, increasing k leads to a decline in the performance of the personalized model. This is because, with a fixed total number of devices, increasing k reduces the number of devices in each group, thereby reducing the total number of samples that the federated model can learn from, which results in a decrease in model performance. In the CIFAR-10 on VGG11 task, as k increases, the accuracy of the personalized model drops from approximately 85% to 78%. It can be observed that under the strategy of randomly selecting 10 devices for collaboration, the model performance outperforms that of $k = 9$ and $k = 12$, as in these two cases, the number of devices in each group (i.e., 8 and 6, respectively) is smaller than the number of collaborators in the random strategy. In another task, the performance of the random strategy is the worst, with an accuracy of about 60%, whereas even with a larger $k = 12$, the average model accuracy reaches 63%. In addition, when $k = 6$, the average accuracy of personalized models reaches its peak of 72% around the 23rd iteration, but then gradually decreases to about 67% as the number of iterations increases. We speculate that this is due to overfitting. To verify this, we examine the average loss of the models after each iteration and find out that the loss on the training set is indeed very small, which confirms our hypothesis.

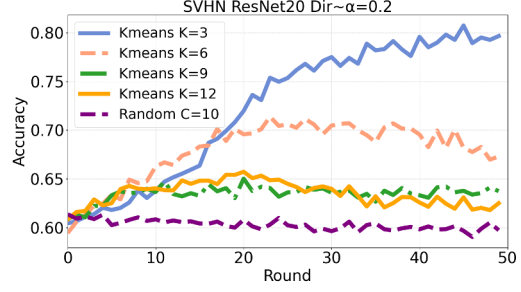
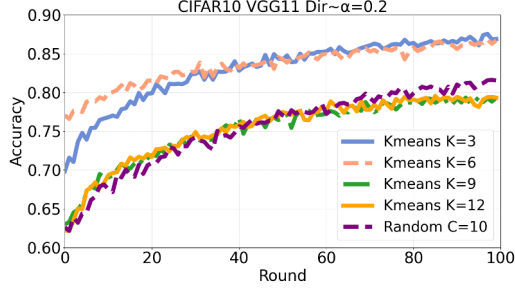


Fig. 8: Performance comparison under different values of k ($\alpha = 0.2$)

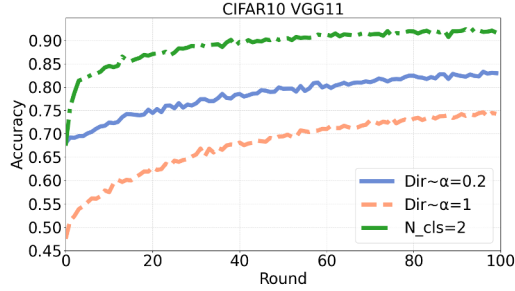


Fig. 9: Performance for heterogeneous devices

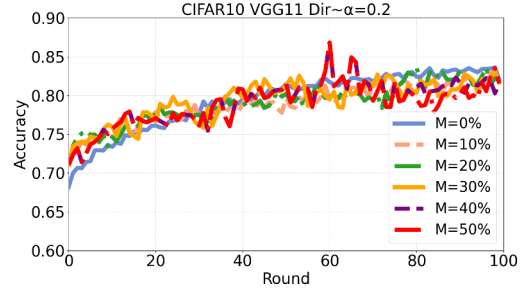


Fig. 10: Performance under different values of m

F. The Impact of Device Dynamics on Performance

In contrast to previous experiments, this set of experiments considers the dynamic nature of the DFL system, where devices may join or leave the system during each iteration. We introduce a parameter m to control the rate of device turnover. In each experiment, 100 devices are randomly selected as initial participants. After each iteration, the set of participants is updated by removing devices with an $m\%$ probability from the current participants and adding devices from the remaining 100 devices with the same probability. As a result, the number of devices participating in each iteration fluctuates around 100. We experiment with different values of m (0, 10, 20, 30, 40, 50) using CIFAR-10 on VGG11, with a pruning rate of 0.5, under Dirichlet distribution with parameters $\alpha = 0.2$.

The trend of the average accuracy of the personalized models obtained by PRISMA as the number of iterations increases, under different values of m , is shown in Figure 9. It can be observed that PRISMA performs robustly in scenarios where devices frequently join and leave the DFL system. Even with 50% of the devices being replaced, there is no significant difference in terms of model convergence speed and final accuracy compared to the static device scenario.

G. The Impact of Device Heterogeneity on Performance

In this experiment, we considered 100 heterogeneous devices, each randomly assigned a pruning rate from three options: 0.3, 0.5, and 0.7. For the task of CIFAR-10 on VGG, we tested PRISAM under three different data distributions, with devices grouped into 5 clusters based on data distribution similarity. Notably, devices within the same group do not

necessarily share the same pruning rate, meaning the federated model aggregation utilized models with different pruning rates.

Figure 10 shows the convergence of personalized models during federated aggregation with mixed pruning rates under three different data distributions. It can be observed that the personalized model performs best under the pathological distribution. However, as the data similarity between groups increases, the performance of the personalized model decreases. This might be because devices struggle to correctly identify other devices with similar data distributions, leading to collaboration with devices that have dissimilar data distributions. Interestingly, we also found that the models obtained with mixed pruning rates outperform those with fixed pruning rates. For example, under the Dirichlet distribution with $\alpha = 0.2$, the model accuracy reaches its highest at 82.1% (pruning rate is 0.5%), while the mixed pruning rate model achieves 82.9%.

VI. CONCLUSION

This paper proposes an efficient personalized DFL framework, PRISAM, based on BN pruning. PRISAM addresses the heterogeneity of data distribution and device. It utilizes BN masks to achieve secure and efficient structured pruning, along with minimizing communication and computational overhead for identifying devices with similar data distributions, which makes it suitable for large-scale cross-device FL scenarios. We validate the effectiveness of PRISAM on three datasets and two models under three types of Non-IID data distributions. In future work, we will examine the performance of PRISAM on more complex tasks and models, as well as distributions, e.g., long-tail mixed with Non-IID.

ACKNOWLEDGMENT

This research was supported in part by the National Natural Science Foundation of China (Grant No. 62472083, 62432008), the Natural Science Foundation of Shanghai (Grant No. 22ZR1400200), the AI-Enhanced Research Program of Shanghai Municipal Education Commission (Grant No. SMEC-AI-DHUZ-01), the RGC RIF grant under contract R6021-20, RGC TRS grant under contract T43-513/23N-2, RGC CRF grants under contracts C7004-22G, C1029-22G and C6015-23G, and RGC GRF grants under contracts 16200221, 16207922 and 16207423.

REFERENCES

- [1] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.
- [2] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.
- [3] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *2019 IEEE 25th international conference on parallel and distributed systems (ICPADS)*, pp. 233–239, IEEE, 2019.
- [4] Y. Liu, S. Chang, Y. Liu, B. Li, and C. Wang, "Fairfed: Improving fairness and efficiency of contribution evaluation in federated learning via cooperative shapley value," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pp. 621–630, IEEE, 2024.
- [5] B. Zhu, S. Chang, G. Liang, H. Zhu, and J. Xu, "Fed-cad: Federated learning with correlation-aware adaptive local differential privacy," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pp. 1–10, IEEE, 2024.
- [6] E. T. Martínez Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Communications Surveys Tutorials*, vol. 25, no. 4, pp. 2983–3013, 2023.
- [7] M. Dai, T. Wang, Y. Li, Y. Wu, L. Qian, and Z. Su, "Digital twin envisioned secure air-ground integrated networks: A blockchain-based approach," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 96–103, 2022.
- [8] R. Dai, L. Shen, F. He, X. Tian, and D. Tao, "Dispf: Towards communication-efficient personalized federated learning via decentralized sparse training," in *International conference on machine learning*, pp. 4587–4604, PMLR, 2022.
- [9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.
- [10] L. Yang, J. Huang, W. Lin, and J. Cao, "Personalized federated learning on non-iid data via group-based meta-learning," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 4, pp. 1–20, 2023.
- [11] H. Jin, D. Bai, D. Yao, Y. Dai, L. Gu, C. Yu, and L. Sun, "Personalized edge intelligence via federated self-knowledge distillation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 567–580, 2022.
- [12] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in neural information processing systems*, vol. 33, pp. 2351–2363, 2020.
- [13] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.
- [14] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu, "Parameterized knowledge transfer for personalized federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10092–10104, 2021.
- [15] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 international joint conference on neural networks (IJCNN)*, pp. 1–9, IEEE, 2020.
- [16] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning: clients clustering for better personalization," *World Wide Web*, vol. 26, no. 1, pp. 481–500, 2023.
- [17] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *IEEE Transactions on Information Theory*, vol. 68, no. 12, pp. 8076–8091, 2022.
- [18] Y.-H. Chou, S. Hong, C. Sun, D. Cai, M. Song, and H. Li, "Grp-fed: Addressing client imbalance in federated learning via global-regularized personalization," in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pp. 451–458, SIAM, 2022.
- [19] Y. Shen, Y. Zhou, and L. Yu, "Cd2-pfed: Cyclic distillation-guided channel decoupling for model personalization in federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10041–10050, 2022.
- [20] Z. Ma, Y. Xu, H. Xu, J. Liu, and Y. Xue, "Like attracts like: Personalized federated learning in decentralized edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1080–1096, 2022.
- [21] Y. Sui, J. Wen, Y. Lau, B. L. Ross, and J. C. Cresswell, "Find your friends: Personalized federated learning with the right collaborators," *arXiv preprint arXiv:2210.06597*, 2022.
- [22] T. Chen, X. Wang, H.-N. Dai, and H. Yang, "A dropout-tolerated privacy-preserving method for decentralized crowdsourced federated learning," *IEEE Internet of Things Journal*, 2023.
- [23] S. Kalra, J. Wen, J. C. Cresswell, M. Volkovs, and H. R. Tizhoosh, "Decentralized federated learning through proxy model sharing," *Nature communications*, vol. 14, no. 1, p. 2899, 2023.
- [24] H. Xie, M. Xia, P. Wu, S. Wang, and K. Huang, "Decentralized federated learning with asynchronous parameter sharing for large-scale iot networks," *IEEE Internet of Things Journal*, vol. 11, no. 21, pp. 34123–34139, 2024.
- [25] Z. Tang, S. Shi, B. Li, and X. Chu, "Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2023.
- [26] 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.
- [27] X. Jiang and C. Borcea, "Complement sparsification: Low-overhead model pruning for federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 8087–8095, 2023.
- [28] H. Huang, L. Zhang, C. Sun, R. Fang, X. Yuan, and D. Wu, "Distributed pruning towards tiny neural networks in federated learning," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pp. 190–201, 2023.
- [29] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.
- [30] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.
- [31] D. Filters' Importance, "Pruning filters for efficient convnets," 2016.
- [32] S. Changpinyo, M. Sandler, and A. Zhmoginov, "The power of sparsity in convolutional neural networks," *arXiv preprint arXiv:1702.06257*, 2017.
- [33] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.
- [34] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: an efficient federated learning framework for heterogeneous mobile clients," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 420–437, 2021.
- [35] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 10374–10386, 2023.
- [36] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.
- [37] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.
- [38] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, et al., "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.