PFA: Privacy-preserving Federated Adaptation for Effective Model Personalization

Bingyan Liu
lby_cs@pku.edu.cn
MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
Beijing, China

Yao Guo*
yaoguo@pku.edu.cn
MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
Beijing, China

Xiangqun Chen
cherry@pku.edu.cn
MOE Key Lab of HCST, Dept of Computer Science, School of EECS, Peking University
Beijing, China

ABSTRACT

Federated learning (FL) has become a prevalent distributed machine learning paradigm with improved privacy. After learning, the resulting federated model should be further personalized to each different client. While several methods have been proposed to achieve personalization, they are typically limited to a single local device, which may incur bias or overfitting since data in a single device is extremely limited. In this paper, we attempt to realize personalization beyond a single client. The *motivation* is that during the FL process, there may exist many clients with similar data distribution, and thus the personalization performance could be significantly boosted if these similar clients can cooperate with each other. Inspired by this, this paper introduces a new concept called *federated adaptation*, targeting at adapting the trained model in a federated manner to achieve better personalization results. However, the key challenge for federated adaptation is that we could not outsource any raw data from the client during adaptation, due to *privacy* concerns. In this paper, we propose **PFA**, a framework to accomplish **P**rivacy-preserving **F**ederated **A**daptation. PFA leverages the sparsity property of neural networks to generate privacy-preserving representations and uses them to efficiently identify clients with similar data distributions. Based on the grouping results, PFA conducts an FL process in a group-wise way on the federated model to accomplish the adaptation. For evaluation, we manually construct several practical FL datasets based on public datasets in order to simulate both the *class-imbalance* and *background-difference* conditions. Extensive experiments on these datasets and popular model architectures demonstrate the effectiveness of PFA, outperforming other state-of-the-art methods by a large margin while ensuring user privacy. We will release our code at: *https://github.com/lebyni/PFA*.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies** → **Neural networks**; • **Security and privacy** → **Privacy protections**.

*Yao Guo is the corresponding author.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449847

KEYWORDS

Decentralized AI, Federated Learning, Neural Networks, Personalization, Privacy

ACM Reference Format:
Bingyan Liu, Yao Guo, and Xiangqun Chen. 2021. PFA: Privacy-preserving Federated Adaptation for Effective Model Personalization. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449847

1 INTRODUCTION

Federated learning (FL) [39, 62] has been proposed and drawn great attention due to its capability to collaboratively train a shared global model under the decentralized data settings. A typical method to implement federated learning is the Federated Averaging (FedAvg) [39], which generates a global model by averaging the local parameters uploaded from each client. During the process, we do not exchange the sensitive raw data in each client and thus protect user privacy. In recent years, there have been extensive applications for deploying FL in practice, such as loan status prediction, health situation assessment, and next-word prediction [21, 59, 60].

Although FL has been proven effective in generating a better federated model, it may not be the optimal solution for each client since the data distribution of clients is inherently non-IID (non-independently identically distribution). Here we believe the distribution not only refers to the statistical heterogeneity (e.g., the number of image category in a certain client) as prior work simulated [33, 39], but also includes the situation where the object is identical while the background is different (e.g. one user may take photos mostly indoors while another mostly outdoors). Under this condition, each client should own a personalized model rather than a global shared model in order to better fit its distinctive data distribution.

Fortunately, the research community has noticed the *data heterogeneity* issue and several methods have been proposed to address the problem. For example, Wang *et al.* [58] accomplished personalization by further fine-tuning the federated model with the local data in each client. Yu *et al.* [61] extended the above work, where the federated model can be personalized via three schemes: fine-tuning, multi-task learning, and knowledge distillation. Although such methods can facilitate personalization to some extent, they have a significant drawback – the personalization process is restricted in a single device, which may introduce some bias or overfitting problem since data in a device is extremely limited. Our **intuition** is that, in the FL process, there may exist many other clients that own similar data distribution to a certain client. If these clients

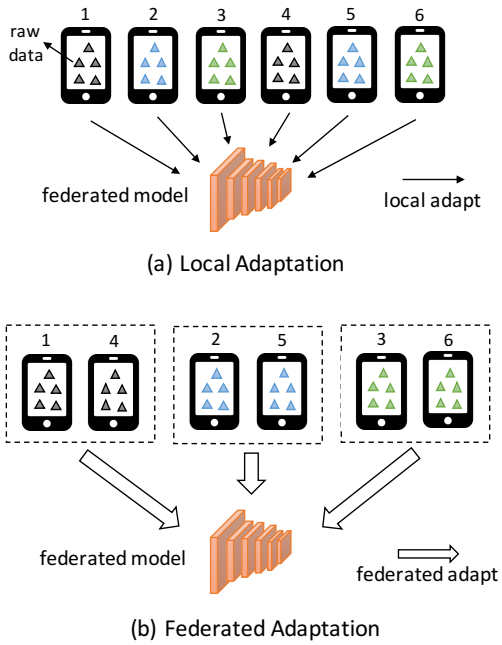

Figure 1: Comparison of two adaptation policies. Different from local adaptation that adapts the federated model with the data of each single client, the proposed federated adaptation approach aims to identify clients with similar data distributions and uses them to collaboratively adapt the federated model. Here the color of the triangle represents the data distribution.

can be aggregated and benefit from each other, the performance will definitely outperform the local adaptation schemes because more valuable data are utilized to personalize the federated model, mitigating the bias or overfitting problem as well as extending the useful knowledge.

Inspired by this, we introduce a new concept called **federated adaptation**, which is defined as *adapting the federated model in a federated manner to achieve better personalization results.* As shown in Figure 1, federated adaptation attempts to use the clients with similar distribution to collaboratively adapt the federated model, rather than just adapting it with the data in a single device. Compared to the traditional federated learning, federated adaptation has the following differences: (1) The adaptation objective is a federated trained model, which means that an FL process should be conducted first before the adaptation begins; (2) Instead of using the whole clients or randomly sampling them as the traditional FL does, in federated adaptation, the federated clients must be selective in order to guarantee the distribution matching.

To the best of our knowledge, there are no existing work that conduct personalization in a federated manner. In order to accomplish federated adaptation, one key challenge is that the raw data in each client cannot be outsourced due to *privacy* concerns. To solve this issue, this paper proposes **PFA**, a prototype framework for achieving personalization via **P**rivacy-preserving **F**ederated **A**daptation. The key idea behind PFA is that *we can leverage the sparsity property of neural networks to generate a privacy-preserving representation which can then be used to replace the raw data for client grouping during the adaptation process.* Specifically, given a federated model, PFA first extracts the client-related sparsity vector as a privacy-preserving representation, and uploads them to the server to distinguish different distributions across clients (**Section 4.1**). By employing Euclidean Distance to measure the similarity between these sparsity vectors, PFA is able to generate a matrix that describes the distribution similarity degree among clients (**Section 4.2**). Based on the matrix, PFA manages to precisely group the clients with similar data distribution and conduct a group-wise FL process to accomplish the adaptation (**Section 4.3**).

Note that existing benchmark datasets of the non-IID setting are designed to simulate the *class-imbalance* scenario [39], which is incomplete to demonstrate the real-world applications. Therefore, we construct several datasets based on some public datasets to simulate both the *class-imbalance* and *background-difference* conditions, in order to better represent the characteristics of practical FL environments (**Section 5.1**).

We evaluate PFA on the constructed datasets and compare it to the FL baseline and three state-of-the-art local adaptation schemes for personalization. The results show that PFA outperforms other methods by up to 8.34%, while ensuring user privacy. Besides, we conduct a number of detailed analyses (e.g., convergence analysis, privacy analysis) to further demonstrate the necessity and effectiveness of PFA.

To summarize, this paper makes the following main contributions:

- **A new idea to achieve better personalization in FL.** We introduce *federated adaptation*, which conducts personalization in a federated manner rather than focusing on a local client.
- **A novel approach to represent data.** We utilize the sparsity property of neural networks to represent the raw data in clients. This representation offers a privacy-preservation way to describe the client-related data distribution accurately.
- **A comprehensive federated adaptation framework.** We propose PFA, a framework to personalize the federated model via privacy-preserving federated adaptation. To the best of our knowledge, this is the first attempt in the literature to explore and study the concept of federated adaptation.
- **Detailed experiments to evaluate the proposed framework.** We conduct extensive experiments based on constructed datasets and state-of-the-art model architectures. The results demonstrate the effectiveness of PFA.

2 BACKGROUND AND TERMINOLOGY

In order to clarify the meaning of specific terms used in this paper, and to help readers get an overall understanding of how federated learning works, we briefly describe some important concepts in the Convolutional Neural Network (CNN) and the workflow of FL.

2.1 Convolutional Neural Network

By default, our work is based on Convolutional Neural Networks (CNNs) [32], which have been widely used in the field of computer

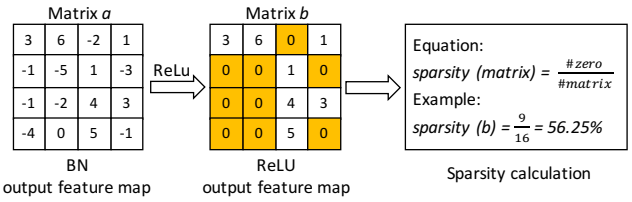

Figure 2: Illustration of the different feature maps and the sparsity calculation.

vision, such as image classification [22, 24, 52], object detection [19, 34] and media interpretation [15, 57]. A typical CNN may include convolutional (Conv) layers, pooling layers, batch normalization (BN) layers, ReLU activation functions and fully connected (FC) layers, each of which represents a unique type of mathematical operations. By feeding an image into a CNN, the output of each layer can be regarded as a type of *feautre map* with a shape of H×W×C, in which H, W, C represent the height, weight, and the number of channels in the feature map. As illustrated in Figure 2, after the processing of the BN and ReLU layer, we can generate the corresponding *BN output feature map* and *ReLU output feature map* (Here we only demonstrate one channel).

Notice that the ReLU layer can turn all negative inputs into zeros, thus making its output highly sparse. Given a ReLU output feature map, for a certain channel, the *sparsity* is defined as the ratio for the number of zero elements in a matrix. For the example in Figure 2, there are 9 zero elements out of 16 elements in the matrix, and thus the *sparsity* is 56.25%. In this way, we can calculate the sparsity value of each channel and the whole feature map will correspond to a sparsity vector. In the following sections, these terms (layer, channel, sparsity *etc.*) will be used to explain our approach.

2.2 Federated Learning

The typical pipeline of FL includes two steps. First, each client trains a model locally with a few epochs and only uploads the parameters/gradients of the model to a central server. Second, the server end coordinates the uploaded information through different aggregation algorithms (e.g., Federated Averaging (FedAvg)), in order to generate a global model that owns the knowledge of each client. The generated global model is finally distributed to each client and may be further trained to iterate the above steps until convergence. During this process, the raw data in each client is not exchanged, thus without leaking the user privacy.

3 GOAL AND CHALLENGE

This section formulates the goal and existing challenges of PFA. The important notations that will be commonly used later are summarized in Table 1.

3.1 Problem Formulation

Starting with a model that is trained by the traditional FL process, PFA first aims to use it to generate a privacy-preserving feature representation vector $R = (R_1, R_2, ..., R_n)$ for clients. Here the R_i denotes the feature representation of the i_{th} client and n is the total number of the clients. The representation vector R is then uploaded to the central server and PFA targets at using it to compare the distribution similarity between clients, generating a similarity matrix S. In terms of S, PFA attempts to schedule clients by partitioning them into different groups, each of which represents a type of data distribution. As a result, each client can benefit from other clients in the group that own data with a similar distribution.

Table 1: Notations used in this paper.

Notation	Explanation
M_f	The federated model
M_p	The personalized model (i.e., our goal)
M_l	The trained local model
R	The privacy-preserving feature representation
S	A similarity matrix among clients
D	The data in a client
E	The element of a specific R_x
N	The number of data in a certain client
sp	The sparsity of a ReLU feature map
n	The number of clients
q	The number of selected channels

Based on these symbols, the final goal of PFA can be defined as follows:

Definition 1. ***(Privacy-preserving federated adaptation)*** *Suppose M_f and M_p^j denote the federated global model and the desirable personalized model of the j_{th} client, respectively. The goal of PFA is to group a series of clients $(j_1, j_2, ..., j_m)$ that own similar data distribution to j, with the help of M_f, R, and S rather than the raw data, in order to adapt M_f in a federated manner and generate M_p^j.*

3.2 Challenges

In this work, we aim at exploring a novel federated adaptation approach to achieve personalization. It is nevertheless non-trivial to accomplish this goal. There are at least three challenges that need to be effectively addressed. We list them as follows.

(1) **How to ensure user privacy in federated adaptation?** As is known to all, privacy protection is the key concern in FL. Similarly, the adaptation process should also pay much attention to the privacy issue. Specifically, the raw data in clients cannot be transferred or inferred during the whole process.
(2) **How to efficiently distinguish clients with different data distributions?** An important point in federated adaptation is that the data of federated clients should come from an identical or similar distribution. Therefore, a similarity metric for client distribution is needed to be developed. Besides, there might be a huge number of clients, which poses a higher demand for an efficient method to accomplish the goal.
(3) **How to accurately and effectively identify and schedule the federated clients?** Unlike the local adaptation in which only one client is needed, for federated adaptation, it is crucial to determine which clients should be aggregated and how they can contribute to each other.

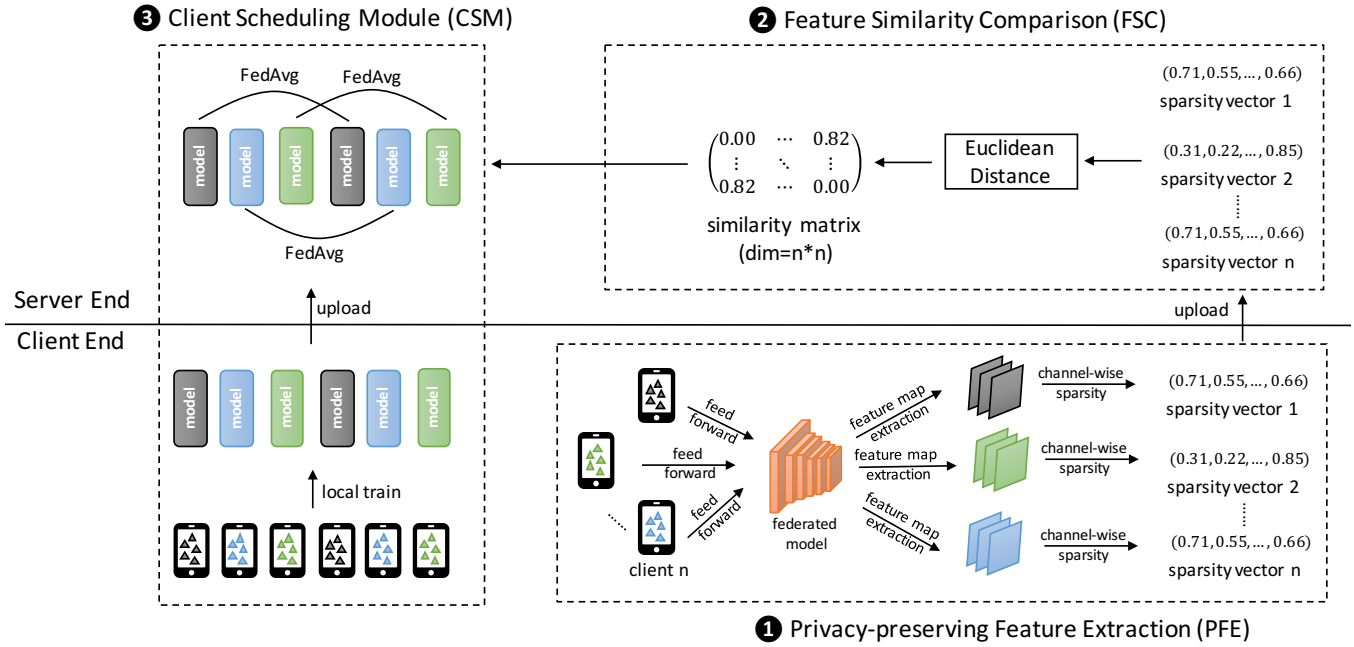

Figure 3: The overview of the proposed framework PFA. Three modules (i.e., PFE, FSC, and CSM) are introduced and sequentially executed to achieve federated adaptation. Concretely, PFE extracts the privacy-preserving representation; FSC uses the representation to generate a similarity matrix between clients and CSM utilizes the matrix to schedule clients by partitioning their corresponding trained models into different groups and conducting group-wise adaptation with the FedAvg algorithm. Note that we do not exchange raw data of clients during the whole process, offering a good privacy protection.

4 OUR FRAMEWORK: PFA

We design and develop PFA, a framework to achieve personalization in a federated manner. Figure 3 depicts the essential modules of the workflow in PFA. Specifically, we introduce three modules to respectively address the aforementioned three challenges:

- **PFE:** a **P**rivacy-preserving **F**eature **E**xtraction module, which takes advantage of the feature map sparsity to represent the distribution property of clients. This module only uses the federated model to conduct a feed forward process, without introducing much computation cost or complex optimizations. The generated sparsity vectors are then uploaded to the server end.
- **FSC:** a **F**eature **S**imilarity **C**omparison module, which employs Euclidean Distance to measure the similarity between the extracted sparsity vectors, in order to generate a similarity matrix that can denote the clients' similarity degree. Besides, we further design a one-client-related similarity vector to form the matrix, so as to achieve efficiency.
- **CSM:** a **C**lient **S**cheduling **M**odule, which partitions the clients into different groups in terms of the similarity matrix generated by **FSC** and implements adaptation in a group-wise manner. Concretely, each client needs to upload its corresponding local trained model and the server selectively groups these models and aggregates them by the FedAvg algorithm.

Besides, we would like to highlight that different modules are conducted in different ends: **PFE** is in the client end; **FSC** is in the server end, and **CSM** requires both of the ends. In the remainder of the section, we describe in detail our approach for implementing each module.

4.1 Privacy-preserving Feature Extraction

The PFE module serves as the first step in the proposed framework. Its objective is to extract a representation for each client that can not only reflect the distribution property but also protect user privacy. A natural idea is to use the feature map extracted from each client as the representation since it is directly associated with the raw data. However, the information of the feature map is easy to be inverted to generate the original raw data [16, 37], which violates the user privacy (details in **Section 5.4**).

In this work, we make an important observation: *by feeding the data of a client into a federated model, its intermediate channel sparsity can express data distribution information of the client.* The insight behind it is that usually the sparsity pattern for specific inputs is unique, which suggests that we can use it as the client representation in order to distinguish different data distributions. Here sparsity refers to the ratio of the number of zero elements in a matrix (details in **Section 2.1**), which has been widely used to accelerate the inference process of Convolutional Neural Networks (CNNs) [7, 47]. Using sparsity as the representation has the following two advantages: (1) Sparsity can be seen as a type of statistical

information, which is intuitively privacy-preserving as it converts the sensitive features of raw data into the simple ratio, making it hard to be inverted (details in **Section 5.4**); (2) Sparsity is smaller than its corresponding feature map, which significantly reduces the communication costs (i.e., uploading them to the server end).

Based on this observation, we attempt to extract the channel-wise sparsity vector as the distribution representation. Specifically, for the i_{th} client, we denote its data as $D_i = (x_1^i, x_2^i, ..., x_N^i)$ and conduct a feed forward process to extract the feature maps. Here N represents the total number of the data in the client. Let $F(x_p^i) \in \mathbb{R}^{H\times W\times C}$ be the feature map extracted from a ReLU layer of the federated model given input $x_p^i \in D_i$. For a channel in the layer, we compute the sparsity by the following equation

$$sp(x_p^i) = \frac{\#zero\ elements}{H \times W} \tag{1}$$

where sp denotes the sparsity. With this equation, we calculate the sparsity of each sample in D_i and average them as the client sparsity for the this channel

$$sp(D_i) = \frac{1}{N}\sum_{k=1}^{N} sp(x_p^i) \tag{2}$$

In this way, we randomly select q channels $(c_1, c_2, ..., c_q)$ from the federated model and extract their corresponding client sparsity to form a sparsity vector, which is considered as the privacy-preserving representation R_i for the i_{th} client. Based on these steps, each client can generate a representation and upload it to the server end for later similarity comparison.

4.2 Feature Similarity Comparison

The FSC module compares the representations (i.e., sparsity vectors) extracted through PFE by computing the distance among them. Here we adopt Euclidean Distance [9] as the metric due to its simplicity and prevalence.

Concretely, for the representation R_i of the i_{th} client and representation R_j of the j_{th} client, their similarity can be measured by

$$sim(R_i, R_j) = \sqrt{\sum_{k=1}^{q}(E_i^k - E_j^k)^2} \tag{3}$$

where E_i^k and E_j^k denote the k_{th} element in R_i and R_j, respectively. Each client representation has q elements since we pick out q channels in above steps. Based on Eq. 3, we calculate the similarity of any of two clients and generate a similarity matrix S, where S_{ij} represents the distribution similarity degree between the i_{th} and the j_{th} client.

Although the above method is effective in comparing the similarity, it may introduce unacceptable computation budgets when the number of clients is huge. For example, we need to calculate $C^2_{100,000}$ times Euclidean Distance if there are 100,000 clients involved in the adaptation phase, which is inefficient and will largely slow the overall adaptation speed.

To overcome the efficiency challenge for fast comparison, we further propose to only calculate the similarity with respect to one client rather than the whole clients. Specifically, we first randomly pick out a representation R_z of the z_{th} client, and then compute the Euclidean Distance between it and other representations, generating a final similarity vector

$$\big(sim(R_z, R_1), sim(R_z, R_2), ..., sim(R_z, R_n)\big) \tag{4}$$

We observe that in fact, this vector is enough to judge the similarity of clients. On one hand, if the value of $sim(R_z, R_g)$ is low, we believe the g_{th} client has similar data distribution to the z_{th} client. On the other hand, if the value of $sim(R_z, R_t)$ and $sim(R_z, R_u)$ is close, the two corresponding clients (the t_{th} and u_{th}) can be considered as sharing the similar data distribution. This judgement is reasonable because generally the option of data distributions is limited (e.g., the number of classes or the background is not infinite in our vision scenario). With the two principles, we can obtain the similarity of any two clients and form the final similarity matrix S. In this way, given 100,000 clients, the total computation budgets are reduced from $C^2_{100,000}$ to 99, 999 times Euclidean Distance calculation, significantly accelerating the comparison process. Experiments in Section 5.5 confirm the effectiveness of the proposed efficient scheme.

Table 2: Statistics of our simulated datasets.

Dataset		Client number	#training sample	#testing sample
Cifar10	*type1*	1,2,3,4,5	5*100	5*100
	type2	6,7,8,9,10	5*100	5*100
	type3	11,12,13,14,15	5*100	5*100
	type4	16,17,18,19,20	5*100	5*100
	type5	21,22,23,24,25	5*100	5*100
Office-Home	*Ar*	1,2,3,4,5	5*291	5*97
	Cl	6,7,8,9,10	5*523	5*174
	Pr	11,12,13,14,15	5*532	5*177
	Rw	16,17,18,19,20	5*522	5*174

4.3 Client Scheduling Module

The client scheduling module aims at picking out a series of clients with similar data distribution and utilizing them to cooperatively personalize the federated model. Thanks to the similarity matrix generated by our FSC module, we are able to easily compare the distribution property of different clients. In practice, there are a large number of clients in the FL process, which indicates that every client might find at least one client with the desirable distribution. Inspired by this, the scheduling module attempts to partition the whole clients into different groups, each of which represents a certain data distribution. As a result, the adaptation process would be operated in each group, getting rid of the influence of other unrelated clients.

Specifically, the pipeline includes the following three steps:

(1) In the client end, each device should first train the federated model for a few epochs, in order to incorporate the client-specific knowledge into the model. The training objective is formulated as follows

$$M_l^i = \arg\min_{M_f} \mathcal{L}(D_i; M_f) \tag{5}$$

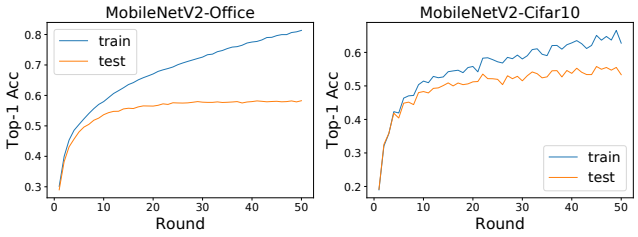

Figure 4: The training process for the two types of datasets on MobileNetV2.

where $\mathcal{L}$ represents the loss calculated by the loss function (e.g., cross-entropy loss) used to optimize parameters. M_l^i is the trained local model for the i_{th} client.

(2) The local trained models $M_l^1, M_l^2, ..., M_l^n$ are then uploaded to the server end and grouped according to the similarity matrix (i.e., if the distribution of two clients is similar, their corresponding local models should be clustered).

(3) We finally conduct the FedAvg algorithm [39], which has been proved effective in aggregating the model knowledge especially for the IID distribution, to accomplish the federated adaptation process. Here in each group, the data distribution can be seen as IID and thus it is desirable to use this algorithm. As shown in Figure 3, models with similar distributions (denoted by colors) collaborate with each other using the FedAvg algorithm, making the resulting model customized to a certain distribution.

The three steps may iterate several times until the aggregated model in each group well fits the corresponding distribution. In this way, each group will finally share an adapted model and this local federated model can be considered as the personalization result. We believe the resulting model is better than the local adapted model because more useful knowledge is aggregated, mitigating the overfitting problem and bias existed in the local adaptation pipeline.

5 EVALUATION

Our evaluation is driven by the following research questions (RQs).

RQ1: What's the performance of PFA? How does it compare to other personalization methods?

RQ2: How effective is the sparsity-based representation in distinguishing clients with different data distributions?

RQ3: Can the extracted sparsity vectors defend the inversion attack and ensure user privacy?

RQ4: How do different extraction strategies of the privacy-preserving representation affect the distribution similarity comparison?

5.1 Experimental Setup

The experimental setup includes the construction of the practical FL datasets, the used models, the implementation details and the compared methods.

Simulated FL datasets. In real-world applications, the distribution divergence can be categorized into two types. The first one is the *class-imbalance* condition, where the statistical information of the data in each client may be extremely different. Take the image classification task as an example, some clients may mainly hold samples of the "cat" class while others may have few "cat" images but a large number of "dog" images. Similar to previous work [39], we simulate this situation by the public Cifar10 dataset [31], which contains 10 different image classes. Specifically, we simulate 25 clients and every 5 clients belongs to a type of distribution. Therefore, totally we have 5 types of distribution, each of which owns two disjoint classes of Cifar10. Considering that the client end may own limited data, we only randomly select 100 samples as the training set and testing set for each client.

In addition to the *class-imbalance* condition, the *background-difference* scenario is also commonly seen in practice. For example, photos taken from different environments sometimes are hard to be classified by neural networks although the main object is identical, which is the focus in the field of domain adaptation [43]. Under this circumstance, we use the Office-Home dataset [55] for simulation. Office-Home contains 15,500 images with four domains: Artistic images (Ar), Clipart images (Cl), Product images (Pr) and Real-World images (Rw). Each domain has 65 identical object categories but different background distributions. Concretely, we divide each domain into 5 parts and each client owns one part. For each part, we further partition it into the training data (80%) and the testing data (20%) since this dataset has no official train/test partition. According to the setting, totally we have 20 clients with 4 types of data distribution. Because the data in each domain of Office-Home is limited, we do not need to implement sampling as we do for Cifar10. We illustrate the detailed statistics in Table 2.

Models. We evaluate the proposed approach on two widely used model architectures: VGGNet [52] and MobileNetV2 [48]. VGGNet is a vanilla convolutional neural network with direct connections across Conv layers. We modify the architecture by replacing the original fully connected layers with a global average pooling layer in order to fit the data scale in our FL setting. MobileNetV2 is a typical convolutional neural network designed for the mobile end, which is suitable for our scenario. For each architecture, we use the smallest variant (i.e., VGG-11 and MobileNetV2-0.25) considering the resource constraint of clients. Besides, we use their pre-trained versions (i.e., trained with the ImageNet dataset [10]) before applying to the actual tasks to accelerate convergence.

PFA implementation. Since it is hard to conduct experiments in real-world FL scenarios, we simulate and operate our experiments in a server that has 4 GeForce GTX 2080Ti GPUs, 48 Intel Xeon CPUs, and 128GB memory. We implement PFA in Python with PyTorch [45] and all the experiments are conducted based on the above datasets and models.

The concrete parameter settings are as follows: In the federated learning and federated adaptation process, the learning rate is set to 0.01, with a momentum of 0.5. The training is conducted for 50 rounds and 30 rounds for the two processes, respectively. In the feature extraction phase, we randomly pick out 30 channels and extract their sparsity to form the sparsity vector. The extraction location is the first ReLU layer of the federated model. Other extraction examples will be further displayed and analyzed in Section 5.5.

Compared methods. We compare the FL baseline and other three personalization methods to our PFA.

Table 3: Office-Home results for the baseline and different personalization methods on VGG-11.

Method	Ar					Cl				
	client1	client2	client3	client4	client5	client6	client7	client8	client9	client10
Baseline	40.21	50.52	59.79	51.55	53.06	42.53	48.85	48.85	48.85	41.81
Fine-tune	43.30	56.70	61.86	54.64	55.10	52.30	53.45	56.32	51.15	49.72
KD	43.30	59.79	63.92	58.76	55.10	50.00	55.75	59.20	54.60	53.11
EWC	43.30	54.64	62.89	58.76	55.10	51.72	54.02	59.20	53.45	52.54
Ours	**43.30**	**61.86**	**63.92**	**60.82**	**60.20**	**59.20**	**59.20**	**63.22**	**59.20**	**53.11**
Method	**Pr**					**Rw**				
	client11	client12	client13	client14	client15	client16	client17	client18	client19	client20
Baseline	73.45	71.75	68.93	76.84	73.89	61.49	66.09	61.49	63.22	68.18
Fine-tune	79.10	77.40	75.71	77.40	77.22	63.22	68.97	65.52	63.22	68.75
KD	79.10	77.40	**78.53**	78.53	78.33	63.22	71.84	64.94	65.52	69.89
EWC	79.10	80.79	76.84	77.40	79.44	63.22	70.69	**65.52**	63.22	69.32
Ours	**81.36**	**80.79**	76.84	**79.10**	**81.67**	**65.52**	**72.99**	64.94	**68.39**	**71.02**

Table 4: Office-Home results for the baseline and different personalization methods on MobileNetV2.

Method	Ar					Cl				
	client1	client2	client3	client4	client5	client6	client7	client8	client9	client10
Baseline	42.27	43.30	49.48	44.33	51.02	51.15	51.15	51.15	47.70	42.37
Fine-tune	43.30	49.48	54.64	44.33	50.00	56.32	56.32	58.05	56.32	50.28
KD	42.27	51.55	52.58	46.39	55.10	56.32	56.32	60.34	55.75	54.24
EWC	43.30	53.61	54.64	47.42	**55.10**	55.75	55.75	60.34	58.62	52.54
Ours	**43.30**	**54.64**	**54.64**	**52.58**	52.00	**62.64**	**62.64**	**60.34**	**63.79**	**57.06**
Method	**Pr**					**Rw**				
	client11	client12	client13	client14	client15	client16	client17	client18	client19	client20
Baseline	71.19	70.62	64.41	69.49	67.22	59.20	61.49	60.92	62.07	61.36
Fine-tune	75.14	74.01	67.23	74.01	73.33	60.34	61.49	60.92	59.77	63.07
KD	73.45	72.88	64.41	75.14	72.78	59.20	62.64	62.07	62.07	65.34
EWC	75.71	74.58	68.36	74.58	71.67	61.49	62.64	62.64	60.92	66.48
Ours	**80.79**	**78.53**	**74.58**	**77.97**	**81.67**	**61.49**	**62.64**	**62.64**	**62.64**	**66.48**

(1) *Baseline.* The global federated trained model can be used to test the performance of all clients, which we consider as the baseline.
(2) *Fine-tuning Adaptation [58].* Fine-tuning technique is a popular paradigm to achieve transfer learning [54]. In the context of FL, this adaptation is used to retrain all parameters of a trained federated model on the participant's local training data.
(3) *KD Adaptation [61].* Knowledge Distillation (KD) [23] extracts information from a "teacher" model into a "student" model. Here we treat the federated model as the teacher and the adapted model as the student in order to implement the knowledge distillation process. The distilled model can be regarded as the personalized model.
(4) *EWC Adaptation [61].* Elastic Weight Consolidation (EWC) [29] is used to overcome the catastrophic forgetting problem [17]. In our scenario, we aim to utilize it to force the federated model to be adapted for the client data while preserving the original important knowledge, with the purpose of mitigating the overfitting problem to some extent.

Table 5: Achieved accuracy (%) of different methods on VGG-11. *Here Ar, Cl, Pr, Rw represent different domains.*

Method	Ar	Cl	Pr	Rw	Avg
Baseline	51.03	46.18	72.97	64.09	58.57
Fine-tune	54.32	52.59	77.37	65.94	62.55
KD	56.17	54.53	78.38	67.08	64.04
EWC	54.94	54.19	78.71	66.39	63.56
Ours	**58.02**	**58.79**	**79.95**	**68.57**	**66.33**

Note that all these methods are conducted in a single device, failing to borrow the useful knowledge existed in other devices.

5.2 RQ1: Overall Results

Before applying personalization, we first need to implement traditional FL to generate a federated model. Towards our simulated FL datasets, we observe that using FL alone cannot achieve a good performance. To give a more direct understanding, we visualize the training process for the two types of datasets on MobileNetV2.

Table 6: Achieved accuracy (%) of different methods on MobileNetV2.

Method	Ar	Cl	Pr	Rw	Avg
Baseline	46.08	48.70	68.59	61.01	56.09
Fine-tune	48.35	55.00	72.74	61.12	59.30
KD	49.58	56.59	71.73	62.26	60.04
EWC	50.81	56.94	72.98	62.83	60.89
Ours	**51.43**	**61.18**	**78.71**	**63.18**	**63.62**

As shown in Figure 4, for Office-Home, the gap between the training and testing accuracy is extremely large, suggesting that the federated model cannot be generalized well. For Cifar10, we can clearly see the instability both in the training and testing phase, which indicates that the model is hard to converge during the FL process. These two phenomena further demonstrate the necessity to personalize the federated model for better performance.

Personalization on Office-Home. Given a federated model, we test its accuracy as the baseline and conduct personalization using 4 methods (i.e., fine-tuning, KD, EWC, and PFA). The detailed results of each client are shown in Table 3 and Table 4. For most clients, the personalization performance (i.e., accuracy) achieved by PFA outperforms other methods by a large margin, both on the VGG-11 and MobileNetV2-0.25 model. Specifically, For VGG-11, PFA demonstrates its superiority on 18 out of 20 clients with up to 6.90% accuracy improvement (for *client6*). For MobileNetV2-0.25, 19 out of 20 clients gain more benefit from the proposed approach compared to the baseline and other personalization methods. Notably, for *client15*, PFA exceeds other methods by 8.34%, bringing a significantly positive effect to the user experience.

In addition, we average the accuracy of each domain and report the domain performance in Table 5 and Table 6. We summarize the following conclusions: (1) The benefits of personalization vary from different domains. For example, personalization contributes less on clients of the *Rw* domain. We believe this is because the *Rw* domain includes more images with general features, which can also be facilitated by other domain data during the FL process although their distributions are different. (2) For the same domain, the model architecture can largely affect the personalization performance of PFA. For instance, the improvement using PFA on VGG-11 is far less than the MobilenetV2-0.25 under the *Pr* domain. Overall, the average improvement is over 2% on both models, which confirms the effectiveness of the proposed approach.

Personalization on Cifar10. Figure 5 illustrates the results on Cifar10. Here we use the accuracy improvement to the baseline as the measurement of the performance. Each point in the figure represents a client model generated by a certain personalization method. As shown in the figure, all the personalization methods can significantly outperform the baseline performance, suggesting that the federated model on the *class-imbalance* setting has greater demand to be personalized. Among these methods, our PFA can achieve consistently better accuracy, especially when the baseline accuracy is low. Besides, it is noteworthy that the KD method can not perform well. The reason may be that this method uses the federated model as a "teacher" while the "teacher" itself is not good enough. In summary, PFA shows its superiority due to federated adaptation.

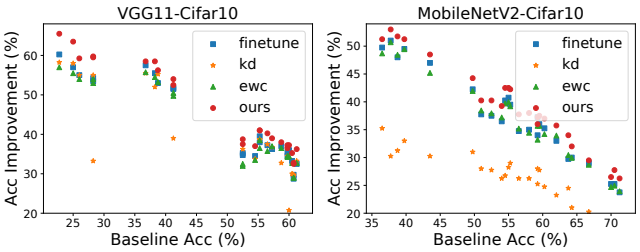

Figure 5: Accuracy improvement achieved by different methods on Cifar10.

Table 7: The number of parameters for uploading.

Item	VGG11	MobileNetV2	10_sp	30_sp	50_sp
#Params	9.23M	0.13M	40B	120B	200B

Table 8: Achieved accuracy (%) on different policies of client selection.

Client number	Federated learning	Random selection	Sparsity-based selection
client1	42.27	39.18	**43.30**
client2	43.30	39.18	**54.64**
client3	49.48	48.45	**54.64**
client4	44.33	47.42	**52.58**
client5	51.02	52.00	**52.00**
client6	51.15	52.30	**62.64**
client7	51.15	51.72	**62.07**
client8	51.15	55.75	**60.34**
client9	47.70	52.87	**63.79**
client10	42.37	50.28	**57.06**
client11	71.19	66.67	**80.79**
client12	70.62	66.67	**78.53**
client13	64.41	63.28	**74.58**
client14	69.49	66.10	**77.97**
client15	67.22	68.89	**81.67**
client16	59.20	58.05	**61.49**
client17	61.49	56.32	**62.64**
client18	60.92	57.47	**62.64**
client19	62.07	60.34	**62.64**
client20	61.36	57.39	**66.48**
Avg	56.09	55.52	**63.62**

Communication costs. Our approach requires uploading some sparsity vectors to the server, which would introduce other communication costs. Table 7 details the number of parameters for uploading. Here the "xx_sp" denotes the size of the sparsity vector. Obviously, compared to the model itself, the parameters of the sparsity information are negligible, indicating that the overall communication costs would not increase much.

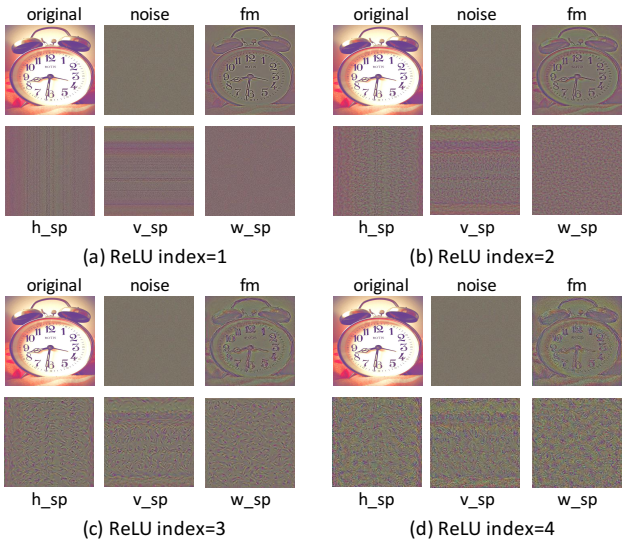

Figure 6: Inversion attack on different properties.

5.3 RQ2: Effectiveness of the Sparsity-based Representation

The sparsity-based representation is a key component that can be used to distinguish clients with different distributions. Thus it is important to assess its effectiveness. Specifically, we evaluate this representation by comparing the following three policies.

- *Federated learning.* We conduct the traditional FL process as the baseline. This also can be considered as selecting the whole clients to implement federated adaptation.
- *Random client selection for federated adaptation.* We randomly pick out clients to conduct federated adaptation, in order to observe the mutual influence among randomly selected clients.
- *Sparsity-based client selection for federated adaptation.* We select clients based on the uploaded sparsity representation. Clients with a high similarity of the representation are aggregated to accomplish federated adaptation.

Here we use the MobileNetV2 on Office-Home as an example and each policy is implemented based on the setting. The detailed accuracy results are summarized in Table 8. From the table, we can clearly see that our sparsity-based selection policy performs best for all the 20 clients by a large margin. On average, the accuracy improvement is 7.52% and 8.10% for federated learning and random selection, respectively, which validates the effectiveness of the extracted sparsity vectors. Besides, we notice that the random selection cannot perform well, even worse than the traditional federated learning on the average performance. This further demonstrates that the federated clients should be selective in the context of the adaptation scenario.

5.4 RQ3: Privacy Analysis

In addition to confirming the effectiveness of the proposed sparsity-based representation, whether this representation can protect user privacy should also be considered. This part provides an analysis with respect to privacy. Concretely, we explore if the sparsity vector can be inverted to generate the original image using the existing attack strategy [16].

We illustrate the results in Figure 6. Four properties (i.e., *fm*, *h_sp*, *v_sp*, *w_sp*) are utilized to implement the inversion attack at different locations (i.e., ReLU index). Here *fm* represents the feature map. *h_sp*, *v_sp*, *w_sp* respectively denote the horizontal-level, vertical-level and whole sparsity of the feature map. Note that we use the sum of the sparsity as the property since a single sparsity value is non-differentiable. From the figure, we can observe that: (1) The *fm* can be easily attacked by inversion since the generated image has abundant features of the original image (e.g., the concrete time in the clock); (2) Only some lines are visualized by inverting the *h_sp* and *v_sp*, which cannot reflect any useful information; (3) With the *w_sp*, attackers fail to inspect any features and the inverted image is just like noises, significantly ensuring the user privacy; (4) The deeper the layer is, the harder we can invert the properties. For example, when it comes to the 4_{th} ReLU index, the generated image from all of the properties tends to be noises. However, even if for the first ReLU layer, it is impossible to invert useful knowledge from the sparsity-based representation, which validates its ability to guarantee the confidentiality of user data.

5.5 RQ4: Influence on Different Extraction Strategies

In aforementioned implementation details, we pick out 30 channels' sparsity from the first ReLU layer as the extracted representation. However, there are other options for the extraction. Therefore, we make an in-depth analysis of different extraction strategies, so as to discover the best solution to facilitate personalization.

Specifically, we first select 8 clients (i.e., 1,2,6,7,11,12,16,17) from the simulated Office-Home dataset and every 2 clients comes from the same distribution. Then for each client, we extract the sparsity vector from each ReLU layer of the federated VGG-11 model. The size (i.e., number of channels) of the vector varies from 10, 20, and 30. Finally, we calculate the Euclidean Distance among these extracted vectors as the metric of client similarity. Here we use *client1* as the base and compare the similarity between it and other clients (i.e., the efficient scheme stated in **Section 4.2**). As shown in Figure 7, the heat map is utilized to visualize the similarity degree, where the darker color represents the higher divergence. We find two interesting phenomena. First, the representation from shallow layers performs better than deep layers. This is reasonable because the sparsity of deep layers is extremely high (sometimes can exceed 90% as illustrated in [7]), making it hard to express the input patterns. Another observation is that more channels contribute to distinguishing clients with different data distributions and in this experiment 30 channels are enough to achieve a good performance.

In summary, the effectiveness of the sparsity depends on the extraction location and quantity. We confirm by experiments that extracting more channels from shallower layers would significantly benefit the final distinguishment.

6 DISCUSSIONS

This section summarizes some limitations of PFA and discusses possible future directions.

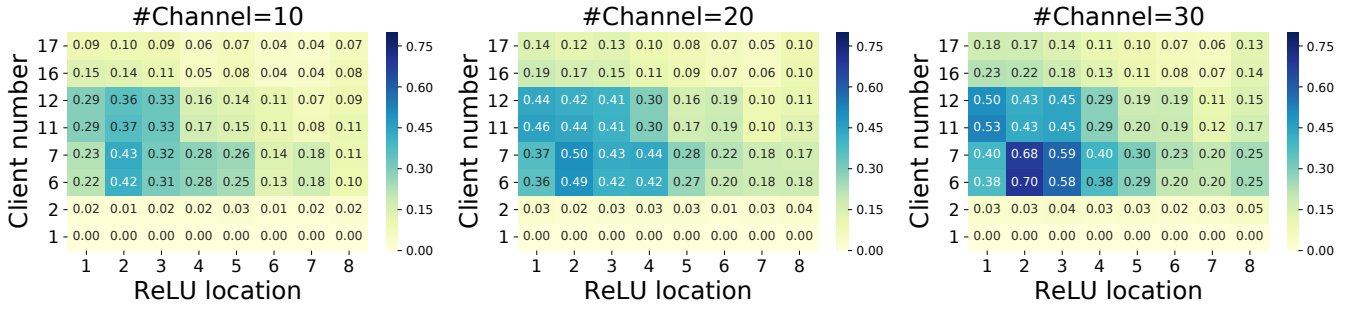

Figure 7: Visualization of the influence on different extraction strategies.

More model architectures. Currently, our framework only targets the CNN-based architecture that is widely used in computer vision tasks. Considering there are many other architectures such as recurrent neural networks (RNNs) and graph convolutional networks (GCNs), it is meaningful to extend our work to cope with these models in the future.

Formal privacy guarantee. Although we have confirmed that the sparsity-based representation cannot be inverted and thus ensures user privacy, it would be better to provide a formal privacy guarantee. For example, the sparsity-based representation can be further added to some noises based on *differential privacy* [12], a commonly used technique to avoid privacy leakage. In our future work, we will look for ways to introduce such protection.

Real-world applications. In this paper, we only evaluate the performance of PFA on simulated datasets and clients. The reason is that it is infeasible to find so many client users and encourage them to conduct our pipeline. However, our simulation design is based on real-world scenarios, which, to some extent, can validate the usefulness of the proposed approach for practical environments.

7 RELATED WORK

Federated Learning and Personalization. Federated learning (FL) enables deep learning models to learn from decentralized data without compromising privacy [28, 39]. Various research directions have been explored to investigate this promising field, such as security analysis for FL [5, 6, 40], efficient communication for FL [3, 25, 30], personalization for FL [26, 38, 51], etc.

This paper focuses on personalization, which stems from the different data distributions of federated clients. There have been a large number of personalization techniques targeting FL. Mansour *et al.* [38] proposed an idea of *user clustering*, where similar clients are grouped together and a separate model is trained for each group. However, this method needs the raw data of each user to implement clustering, which is infeasible due to privacy concerns. In order to guarantee privacy, Wang *et al.* [58] proposed to utilize transfer learning to achieve personalization, where some or all parameters of a trained federated model are re-learned (i.e., fine-tuned) on the local data, without any exchange to the user data. Similarly, Jiang *et al.* [26] also conducted fine-tuning to personalize the federated model but this model was generated by a meta-learning [14] way. Yu *et al.* [58] further extended prior work and systematically evaluated the performance of three personalization methods (i.e., fine-tuning, multi-task learning, knowledge distillation). Different from the above approaches that are implemented in a local client, our framework manages to utilize more useful knowledge existed in other clients by *federated adaptation*, getting rid of the overfitting problem or training bias during the personalization process. Meanwhile, we do not upload any raw data to the server, ensuring user privacy.

Sparsity of CNNs. Researchers have proposed to take advantage of sparsity in the activation maps to speed up CNNs [20, 27, 35, 44, 49]. The main observation is that the Rectified linear unit (ReLU) activation often contains more than 50% zeros on average. Inspired by this, both hardware-based and software-based convolution algorithms are proposed to exploit the input sparsity. In addition, there are some methods aimed at predicting sparsity in order to skip computation of those unimportant activation spaces [2, 11, 13, 47, 53]. In our work, we attempt to use sparsity to distinguish clients with diverse distributions, a completely different utilization to existing work targeting at acceleration.

Privacy Protection of User Data. As people are paying more and more attention to their sensitive data, policies such as the General Data Protection Regulation (GDPR) [56] and Health Insurance Portability and Accountability Act (HIPAA) [4] have been proposed to formally guarantee user privacy. From the technical view, data can be protected by the following two approaches. On one hand, we can rely on secure multiparty computation (SMC) to jointly compute a public function without mutually revealing private inputs by executing cryptographic protocols [8, 42, 46]. On the other hand, differential privacy (DP) [1, 12, 18] can be adopted by adding noises to the data, with the purpose of obfuscating the privacy properties and avoiding the user attributes to be inferred [36, 50]. However, SMC requires massive computation power and communication costs, which is unacceptable to the client end. For DP, our sparsity-based representation can be easily added to noises to satisfy the principle of DP, and we leave this implementation in the future work. In addition, Meurisch *et al.* [41] presented a new decentralized and privacy-by-design platform, in which different approaches to privacy were adopted to benefit personalization. We believe our approach can also be integrated into the platform to further facilitate the personalization research.

8 CONCLUSION

In this paper, we propose the idea of federated adaptation, which extends existing personalization techniques restricted in a single device. We design and implement a framework named PFA to accomplish federated adaptation in a privacy-preserving manner. PFA takes advantage of the sparsity vector to compare and schedule clients, in order to identify suitable federated clients to conduct the adaptation. Experiments on our simulated datasets and clients demonstrate the effectiveness of PFA, outperforming other personalization methods while ensuring privacy. To the best of our knowledge, this is the first work to study and achieve federated adaptation. We hope our work can provide a new angle to personalization-related research in the FL community.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback. This work was partly supported by the National Key Research and Development Program (2016YFB1000105) and the National Natural Science Foundation of China (61772042).

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Vahideh Akhlaghi, Amir Yazdanbakhsh, Kambiz Samadi, Rajesh K Gupta, and Hadi Esmaeilzadeh. 2018. Snapea: Predictive early activation for reducing computation in deep convolutional neural networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 662–673.

[3] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*. 1709–1720.

[4] George J Annas. 2003. HIPAA regulations—a new era of medical-record privacy?

[5] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[6] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*. PMLR, 634–643.

[7] Shijie Cao, Lingxiao Ma, Wencong Xiao, Chen Zhang, Yunxin Liu, Lintao Zhang, Lanshun Nie, and Zhi Yang. 2019. Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11216–11225.

[8] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*. Springer, 643–662.

[9] Per-Erik Danielsson. 1980. Euclidean distance mapping. *Computer Graphics and image processing* 14, 3 (1980), 227–248.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[11] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. 2017. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5840–5848.

[12] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.

[13] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. 2017. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1039–1048.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* (2017).

[15] Thomas Forgione, Axel Carlier, Géraldine Morin, Wei Tsang Ooi, Vincent Charvillat, and Praveen Kumar Yadav. 2018. An Implementation of a DASH Client for Browsing Networked Virtual Environment. In *Proceedings of the 26th ACM international conference on Multimedia*. 1263–1264.

[16] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1322–1333.

[17] Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* 3, 4 (1999), 128–135.

[18] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[20] Benjamin Graham and Laurens van der Maaten. 2017. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307* (2017).

[21] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[25] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al. 2019. Communication-efficient distributed sgd with sketching. In *Advances in Neural Information Processing Systems*. 13144–13154.

[26] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. 2019. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488* (2019).

[27] Patrick Judd, Alberto Delmas, Sayeh Sharify, and Andreas Moshovos. 2017. Cnvlutin2: Ineffectual-activation-and-weight-free deep neural network computing. *arXiv preprint arXiv:1705.00125* (2017).

[28] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[30] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[31] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[33] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[34] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[35] Bingyan Liu, Yao Guo, and Xiangqun Chen. 2019. WealthAdapt: A general network adaptation framework for small data tasks. In *Proceedings of the 27th ACM International Conference on Multimedia*. 2179–2187.

[36] Bingyan Liu, Yuanchun Li, Yunxin Liu, Yao Guo, and Xiangqun Chen. 2020. PMC: A Privacy-preserving Deep Learning Model Customization Framework for Edge Computing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–25.

[37] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5188–5196.

[38] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).

[39] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[40] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.

[41] Christian Meurisch, Bekir Bayrak, and Max Mühlhäuser. 2020. Privacy-preserving AI Services Through Data Decentralization. In *Proceedings of The Web Conference 2020*. 190–200.

[42] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 113–124.

[43] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[44] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W Keckler, and William J Dally. 2017. Scnn: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH Computer Architecture News* 45, 2 (2017), 27–40.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*. 8026–8037.

[46] Vincent Primault, Vasileios Lampos, Ingemar Cox, and Emiliano De Cristofaro. 2019. Privacy-Preserving Crowd-Sourcing of Web Searches with Private Data Donor. In *The World Wide Web Conference*. 1487–1497.

[47] Mengye Ren, Andrei Pokrovsky, Bin Yang, and Raquel Urtasun. 2018. Sbnet: Sparse blocks network for fast inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8711–8720.

[48] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[49] Shaohuai Shi and Xiaowen Chu. 2017. Speeding up convolutional neural networks by exploiting the sparsity of rectifier units. *arXiv preprint arXiv:1704.07724* (2017).

[50] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[51] Khe Chai Sim, Petr Zadrazil, and Françoise Beaufays. 2019. An investigation into on-device personalization of end-to-end automatic speech recognition models. *arXiv preprint arXiv:1909.06678* (2019).

[52] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[53] Mingcong Song, Jiechen Zhao, Yang Hu, Jiaqi Zhang, and Tao Li. 2018. Prediction based execution on deep neural networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 752–763.

[54] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. 2016. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging* 35, 5 (2016), 1299–1312.

[55] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5018–5027.

[56] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* (2017).

[57] Vedran Vukotić, Christian Raymond, and Guillaume Gravier. 2016. Multimodal and crossmodal representation learning from textual and visual features with bidirectional deep neural networks for video hyperlinking. In *Proceedings of the 2016 ACM workshop on Vision and Language Integration Meets Multimedia Fusion*. 37–44.

[58] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. 2019. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252* (2019).

[59] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[60] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[61] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. 2020. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758* (2020).

[62] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).