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 Privacy-preserving Federated Adaptation. 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.

KEYWORDS

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

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
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 Privacy-preserving Federated Adaptation. 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 4.5).

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
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 channel can be regarded as a type of feature 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.

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

**Definition 1.** *(Privacy-preserving federated adaptation)* Suppose \( M_f \) and \( M_p \) denote the federated global model and the desirable personalized model of the \( 3\)-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 \( M_p \).

### 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. **(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. **(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. **(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.

### Table 1: Notations used in this paper.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_f )</td>
<td>The federated model</td>
</tr>
<tr>
<td>( M_p )</td>
<td>The personalized model (i.e., our goal)</td>
</tr>
<tr>
<td>( M_l )</td>
<td>The trained local model</td>
</tr>
<tr>
<td>( R )</td>
<td>The privacy-preserving feature representation</td>
</tr>
<tr>
<td>( S )</td>
<td>A similarity matrix among clients</td>
</tr>
<tr>
<td>( D )</td>
<td>The data in a client</td>
</tr>
<tr>
<td>( E )</td>
<td>The element of a specific ( R )</td>
</tr>
<tr>
<td>( N )</td>
<td>The number of data in a certain client</td>
</tr>
<tr>
<td>( sp )</td>
<td>The sparsity of a ReLU feature map</td>
</tr>
<tr>
<td>( n )</td>
<td>The number of clients</td>
</tr>
<tr>
<td>( q )</td>
<td>The number of selected channels</td>
</tr>
</tbody>
</table>
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 Privacy-preserving Feature Extraction 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 Feature Similarity Comparison 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 Client Scheduling Module, 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 $x_p^i$. For a channel in the layer, we
compute the sparsity by the following equation

$$sp(x_p^i) = \frac{\# \text{zero elements}}{H \times W}$$  (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 this channel

$$sp(D_i) = \frac{1}{N} \sum_{p}^N sp(x_p^i)$$  (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 represen-
tation $R_j$ of the $j_{th}$ client, their similarity can be measured by

$$sim(R_i, R_j) = \sqrt{\frac{1}{q} \sum_{k=1}^{q} (E_i^k - E_j^k)^2}$$  (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 $q$ chan-
nels 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 similar-
ity, it may introduce unacceptable computation budgets when the
number of clients is huge. For example, we need to calculate $C_{100,000}^2$
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, gen-
erating a final similarity vector

$$(sim(R_z, R_1), sim(R_z, R_2), ..., sim(R_z, R_n))$$  (4)

We observe that in fact, this vector is enough to judge the similarity
of clients. On one hand, if the value of $sim(R_z, R_{t})$ 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 above, 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
$C_{100,000}^2$ to 99,999 times Euclidean Distance calculation, signifi-
cantly accelerating the comparison process. Experiments in Section
5.5 confirm the effectiveness of the proposed efficient scheme.

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

$$M^*_l = \arg \min_{M_l} L(D_l; M_l)$$  (5)
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.

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

where $L$ represents the loss calculated by the loss function (e.g., cross-entropy loss) used to optimize parameters. $M_i$ is the trained local model for the $i$th client.

(2) The local trained models $M_1, M_2, ..., M_n$ 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 over-fitting 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.
Table 3: Office-Home results for the baseline and different personalization methods on VGG-11.

<table>
<thead>
<tr>
<th>Method</th>
<th>Ar</th>
<th>Cl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>client1</td>
<td>client2</td>
</tr>
<tr>
<td>Baseline</td>
<td>40.21</td>
<td>50.52</td>
</tr>
<tr>
<td>Fine-tune</td>
<td>43.30</td>
<td>56.70</td>
</tr>
<tr>
<td>KD</td>
<td>43.30</td>
<td>59.79</td>
</tr>
<tr>
<td>EWC</td>
<td>43.30</td>
<td>54.64</td>
</tr>
<tr>
<td>Ours</td>
<td>43.30</td>
<td>61.86</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Pr</th>
<th>Rw</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>client1</td>
<td>client2</td>
</tr>
<tr>
<td>Baseline</td>
<td>73.45</td>
<td>71.75</td>
</tr>
<tr>
<td>Fine-tune</td>
<td>79.10</td>
<td>77.40</td>
</tr>
<tr>
<td>KD</td>
<td>79.10</td>
<td>77.40</td>
</tr>
<tr>
<td>EWC</td>
<td>79.10</td>
<td>80.79</td>
</tr>
<tr>
<td>Ours</td>
<td>81.36</td>
<td>80.79</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Ar</th>
<th>Cl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>client1</td>
<td>client2</td>
</tr>
<tr>
<td>Baseline</td>
<td>42.27</td>
<td>43.30</td>
</tr>
<tr>
<td>Fine-tune</td>
<td>43.30</td>
<td>49.48</td>
</tr>
<tr>
<td>KD</td>
<td>42.27</td>
<td>51.55</td>
</tr>
<tr>
<td>EWC</td>
<td>43.30</td>
<td>53.61</td>
</tr>
<tr>
<td>Ours</td>
<td>43.30</td>
<td>54.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Pr</th>
<th>Rw</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>client1</td>
<td>client2</td>
</tr>
<tr>
<td>Baseline</td>
<td>71.19</td>
<td>70.62</td>
</tr>
<tr>
<td>Fine-tune</td>
<td>75.14</td>
<td>74.01</td>
</tr>
<tr>
<td>KD</td>
<td>73.45</td>
<td>72.88</td>
</tr>
<tr>
<td>EWC</td>
<td>75.71</td>
<td>74.58</td>
</tr>
<tr>
<td>Ours</td>
<td>80.79</td>
<td>78.53</td>
</tr>
</tbody>
</table>

(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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Ar</th>
<th>Cl</th>
<th>Pr</th>
<th>Rw</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>client1</td>
<td>client2</td>
<td>client3</td>
<td>client4</td>
<td>client5</td>
</tr>
<tr>
<td>Baseline</td>
<td>51.03</td>
<td>46.18</td>
<td>72.97</td>
<td>64.09</td>
<td>58.57</td>
</tr>
<tr>
<td>Fine-tune</td>
<td>54.32</td>
<td>52.59</td>
<td>77.37</td>
<td>65.94</td>
<td>62.35</td>
</tr>
<tr>
<td>KD</td>
<td>56.17</td>
<td>54.53</td>
<td>78.38</td>
<td>67.08</td>
<td>64.04</td>
</tr>
<tr>
<td>EWC</td>
<td>54.94</td>
<td>54.19</td>
<td>78.71</td>
<td>66.39</td>
<td>63.56</td>
</tr>
<tr>
<td>Ours</td>
<td>58.02</td>
<td>58.79</td>
<td>79.95</td>
<td>68.37</td>
<td>66.33</td>
</tr>
</tbody>
</table>

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.
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 cannot 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.

**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.

---

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Ar</th>
<th>Cl</th>
<th>Pr</th>
<th>Rw</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>46.08</td>
<td>48.70</td>
<td>68.59</td>
<td>61.01</td>
<td>56.09</td>
</tr>
<tr>
<td>Fine-tune</td>
<td>48.35</td>
<td>53.00</td>
<td>72.74</td>
<td>61.12</td>
<td>59.30</td>
</tr>
<tr>
<td>KD</td>
<td>49.58</td>
<td>56.59</td>
<td>71.73</td>
<td>62.26</td>
<td>60.04</td>
</tr>
<tr>
<td>EWC</td>
<td>50.81</td>
<td>56.94</td>
<td>72.98</td>
<td>62.83</td>
<td>60.89</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>51.43</strong></td>
<td><strong>61.18</strong></td>
<td><strong>78.71</strong></td>
<td><strong>63.18</strong></td>
<td><strong>63.62</strong></td>
</tr>
</tbody>
</table>

**Table 7: The number of parameters for uploading.**

<table>
<thead>
<tr>
<th>Item</th>
<th>VGG11</th>
<th>MobileNetV2</th>
<th>10_sp</th>
<th>30_sp</th>
<th>50_sp</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Params</td>
<td>9.23M</td>
<td>0.13M</td>
<td>40B</td>
<td>120B</td>
<td>200B</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Client number</th>
<th>Federated learning</th>
<th>Random selection</th>
<th>Sparsity-based selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>client1</td>
<td>42.27</td>
<td>39.18</td>
<td>43.30</td>
</tr>
<tr>
<td>client2</td>
<td>43.30</td>
<td>39.18</td>
<td>54.64</td>
</tr>
<tr>
<td>client3</td>
<td>49.48</td>
<td>48.45</td>
<td>54.64</td>
</tr>
<tr>
<td>client4</td>
<td>44.33</td>
<td>47.42</td>
<td>52.58</td>
</tr>
<tr>
<td>client5</td>
<td>51.02</td>
<td>52.00</td>
<td>52.00</td>
</tr>
<tr>
<td>client6</td>
<td>51.15</td>
<td>52.30</td>
<td>62.64</td>
</tr>
<tr>
<td>client7</td>
<td>51.15</td>
<td>51.72</td>
<td>62.07</td>
</tr>
<tr>
<td>client8</td>
<td>51.15</td>
<td>55.75</td>
<td>60.34</td>
</tr>
<tr>
<td>client9</td>
<td>47.70</td>
<td>52.87</td>
<td>63.79</td>
</tr>
<tr>
<td>client10</td>
<td>42.37</td>
<td>50.28</td>
<td>57.06</td>
</tr>
<tr>
<td>client11</td>
<td>71.19</td>
<td>66.67</td>
<td>80.79</td>
</tr>
<tr>
<td>client12</td>
<td>70.62</td>
<td>66.67</td>
<td>78.53</td>
</tr>
<tr>
<td>client13</td>
<td>64.41</td>
<td>63.28</td>
<td>74.58</td>
</tr>
<tr>
<td>client14</td>
<td>69.49</td>
<td>66.10</td>
<td>77.97</td>
</tr>
<tr>
<td>client15</td>
<td>67.22</td>
<td>68.89</td>
<td>81.67</td>
</tr>
<tr>
<td>client16</td>
<td>59.20</td>
<td>58.05</td>
<td>61.49</td>
</tr>
<tr>
<td>client17</td>
<td>61.49</td>
<td>56.32</td>
<td>62.64</td>
</tr>
<tr>
<td>client18</td>
<td>60.92</td>
<td>57.47</td>
<td>62.64</td>
</tr>
<tr>
<td>client19</td>
<td>62.07</td>
<td>60.34</td>
<td>62.64</td>
</tr>
<tr>
<td>client20</td>
<td>61.36</td>
<td>57.39</td>
<td>66.48</td>
</tr>
<tr>
<td>Avg</td>
<td>56.09</td>
<td>55.52</td>
<td>63.62</td>
</tr>
</tbody>
</table>
The sparsity-based representation is a key component that can be privacy should also be considered. This part provides an analysis based representation, whether this representation can protect user 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 \( \text{client}_1 \) 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 distinguishing.

6 DISCUSSIONS

This section summarizes some limitations of PFA and discusses possible future directions.
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, 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. Experiments on our simulated datasets and clients takes advantage of the sparsity vector to compare and schedule device. We design and implement a framework named PFA to achieve federated adaptation.

We hope our work can provide a new angle to personalization methods while ensuring privacy. To the best of our knowledge, this is the first work to study and achieve federated adaptation.

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).

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


