Configure Your Federation: Hierarchical Attention-enhanced Meta-Learning Network for Personalized Federated Learning

YUJIA GAO, PENGFEI WANG, LIANG LIU, CHI ZHANG, and HUADONG MA,
Beijing University of Posts and Telecommunications, China

Federated learning, as a distributed machine learning framework, enables clients to conduct model training without transmitting their data to the server, which is used to solve the dilemma of data silos and data privacy. It can work well on clients having similar data characteristics and distribution. However, it has some limitations where the dataset of clients may be different in distribution, quantity, and concept in many application scenarios. Personalized federated learning is a new federated learning paradigm that aims to guarantee client personalized models' effectiveness when collaborating with the cloud server. Intuitively, providing further facilitated collaborations for the clients with similar data characteristics and distribution can benefit personalized model building. However, due to the invisibility of client data, it is challenging to extract client characteristics and define collaborative relationships among them from a fine-grained view. Moreover, a reasonable collaborative training approach needs to be designed for a distributed server–client framework. In this article, we design a Hierarchical Attention-enhanced Meta-learning Network (HAM) to address this issue. The main advantage of HAM is that it utilizes the meta-learning approach of taking model parameters as features and learns to learn an extra model for each client to analyze similarities according to their local dataset automatically. According to its two-layers framework, HAM can reasonably achieve a tradeoff between clients' personality and commonality and provides a hybrid model with useful information from all clients. Considering there are two networks (HAM and base network) that need to learn for each client during the federated training process, we then provide an alternative learning approach to train them in an end-to-end fashion. To further clarify the approach, we describe the personalized federated learning settings framework as FedHAM where the HAM network is distributed deployed in each client. Extensive experiments based on two datasets prove that our method outperforms state-of-the-art baselines under different evaluation metrics.

CCS Concepts: • **Computing methodologies** → *Cooperation and coordination;*

Additional Key Words and Phrases: Distributed computing, federated learning, deep learning

This work is supported in part by the National Natural Science Foundation of China (61932013, 62061146002, 62225204), and the BUPT Excellent Ph.D. Students Foundation (CX2021308).
Authors' address: Y. Gao, P. Wang, L. Liu, C. Zhang, and H. Ma, Beijing University of Posts and Telecommunications, 10 Xitucheng Road, Beijing, China, 100876; emails: {gaoyujia, wangpengfei, liangliu, zhangchi, mhd}@bupt.edu.cn.

2157-6904/2023/06-ART63 $15.00
https://doi.org/10.1145/3591362

ACM Reference format:
Yujia Gao, Pengfei Wang, Liang Liu, Chi Zhang, and Huadong Ma. 2023. Configure Your Federation: Hierarchical Attention-enhanced Meta-Learning Network for Personalized Federated Learning. *ACM Trans. Intell. Syst. Technol.* 14, 4, Article 63 (June 2023), 24 pages.
https://doi.org/10.1145/3591362

1 INTRODUCTION

Federated learning [32], as a distributed machine learning framework, is used to solve the dilemma of data silos and data privacy. By its distributed structure, federated learning runs collaboratively among a set of clients while preserving their privacy. Thus collaborative clients can achieve better learning performance than individuals working alone. Meanwhile, the controllable transmission interval adopted by federated learning techniques can reduce the communication load and reduce the training latency for time-sensitive tasks, which is beneficial various applications, e.g., medical monitoring [5] and traffic prediction [27]. Traditional settings of federated learning are appropriate for clients whose dataset have similar characteristics and distribution. However, in many scenarios, datasets of different clients always have different distributions, quantities, and concepts [22], which significantly limit the performance of models trained under federated settings. These make the coarse global collaboration cannot achieve good performance for individual clients without considering their data properties. Therefore, it is essential to train personalized models for each client in parallel and achieve global collaborative training.

Recently, various models are designed to incorporate personalized information into federated learning [12, 13]. Personalized federated learning [22] is a new federated learning paradigm, which aims to guarantee the effectiveness of clients' personalized models when collaborating with the cloud server. The commonly used method is to take a unified value (e.g., global model) as the initial value and fine-tune it during the local training process to achieve adaptation to clients [8, 31].

These works lack flexibility because a single model cannot be applied to all clients, so it cannot mitigate the heterogeneity problem. At the same time, the coarse fusion of models also affects model performance. We explore a better approach to provide different collaborative relationships for each client. Specifically, this approach facilitate collaboration among clients with similar characteristics and data distributions, and alienating collaboration among clients with widely varying characteristics. Therefore, *how to automatically identify similar clients for a given client and facilitate their collaborations from a fine-grained review* is a challenging and unresolved problem under federated settings. For a distributed server-client framework, *a reasonable collaborative training approach* needs to be designed as well.

To address these issues, in this article, we propose a **Hierarchical Attention-enhanced Meta-learning Network (HAM)** for personalized federated learning, which aims to resolve the challenge using meta-learning method. It takes model parameters of the client as features and automatically learns a meta-model to analyze similarities among various clients for better performances. In this way, client privacy can be protected by only transmitting model parameters. Simultaneously, local data can be used for meta-model and local model training, which solves the lack of supervised signals.

Specifically, HAM has two layers. Given a target client, in the first layer, HAM runs an attention mechanism to explicitly analyze similarities between model parameter from target client and other clients. According to the learned attention scores, HAM constructs a new model parameter , which is named as the attention-enhance model parameter, by aggregating local models from other clients according to the assigned weighted scores. In the second layer, HAM further aggregates its local model parameter, attention-enhanced model, and the global model parameters to achieve a

tradeoff between personality and commonality. Through such a meaningful design, we can enhance the flexibility of training and meet the accurate decision-making requirements from decentralized data.

Considering privacy preserving between different clients and expecting clients with similar model parameters are encouraged to have robust collaborations, we further design a personalized federated framework, **FedHAM**, where HAM network is distributed deployed in each client. The clients need to train both the HAM network and the base network, where the base network is the backbone network used by the client to train the local model. To reduce the extra computational effort introduced by HAM, we design a specialized training approach. First, we train the base network for each client in a common federated learning way, until local models of each client achieve stability (loss no longer decreases), or reaches a threshold of the communication rounds. Then, we add our HAM to the training process under federated settings. Concretely, we cluster the client's local model parameters on the server, and divide clients into multiple clusters according to their model parameters. The clustering procedure enables coarse grouping of clients so that training can be performed without transferring all model parameters, which reduces the additional communication overhead caused by HAM. Each communication round, the server sends the global model parameters and the set of model parameters in the cluster corresponding to the target client. HAM takes these model parameters as input and uses the client's local data for training. Meanwhile, considering that clients need to train both the HAM and the client's base network, we train two networks using an alternative learning approach guided by the client's objective function.

Extensive experiments on a benchmark dataset and a real-world dataset demonstrate the superior performance of our method. Compared with the state-of-the-art baselines, our method improves the model accuracy performance by at least 3.39% and 2.45%, in two datasets, respectively. Meanwhile, to prove the effectiveness of the hierarchical structure of HAM, we provide an ablation study and a visualization of different layers in HAM.

Finally, to discuss the scalability of FedHAM in dealing with data privacy attacks, we summarize the forms of attacks that the approach may encounter and provide three privacy protection schemes that can be combined with FedHAM.

In total, the contributions of our work are as follows:

- We formalize the personalized federated learning problem into a distributed meta-learning task, and design a novel Hierarchical Attention-enhanced Meta-learning Network to solve the local model personalization problem.
- By treating model parameters as features, we use the attention mechanism to automatically analyze clients' similarities from the parameters level. An alternative learning approach is further applied to enhance the stability and flexibility of training.
- Extensive experiments with different inference tasks show that our method outperforms state-of-the-art baselines and can be applied in many personalized modeling scenarios with distributed data.

The remainder of the article is organized as follows. Section 2 reviews the related works. Section 3 gives the problem formalization of our methods. In Section 4, we introduce the network structure and implement our hierarchical attention-enhanced meta-learning network. In Section 5, we describe the framework overview and training process of FedHAM. Then, we evaluate the performance of our method with the state-of-the-art baselines in Section 6. A discussion of how FedHAM ensures data privacy is given in Section 7. Finally, a brief conclusion and future work are provided in Section 8.

2 RELATED WORK

In this section, we review two research areas related to our work: personalized federated learning and meta-learning.

Personalized Federated Learning. Personalized federated learning aims to train each client by maintaining its personalized model when collaborating with the cloud server. In general, there are three significant categories of personalized federated learning [22], which are local fine-tuning methods, model regularization methods, and multi-task learning methods.

In local fine-tuning, each client receives a global model and tunes it using its local data and several gradient descent steps. For example, PMF [13] designs two personal adaptors (personal bias, personal filter) for higher layers in the user's local model, which can be fine-tuned with personal information. FedPer [3] proposes a base + personalization method, which only trains the base layer collaboratively. FedRep [11] proposes an algorithm for learning a shared data representation across clients and unique local heads for each client. FedBABU [36] updates the body of the global model and fine-tune the head of models for personalization during the evaluation process. In model regularization, the authors of Reference [15] add a regularization term on the distance of local and global models, and use a mixing parameter to control the degree between them. The authors of Reference [41] propose a knowledge distillation way to achieve personalization, where they apply the regularization on the predictions between the local and global model. These methods use a unified global model for personalization and cannot provide flexible personalized modeling for a wide range of potential tasks on heterogeneous data clients.

Federated Multi-task Learning [43] considers the optimization of each client as a new task. It tackles communication constraints, stragglers, and fault tolerance, which focus on the convex model. However, due to its rigid requirement of strong duality, this method is no longer guaranteed applicable when clients adopt non-convex deep learning models. The authors of Reference [19] use an attention-including function to measure the difference between model parameters. Although it models a pairwise collaboration among clients, it still has limitations in depicting dynamic collaboration between clients.

Meta-Learning. Meta-learning has been an active research area in recent years. Due to its promise to be able to generalize well given limited amounts of training data, it is widely applied in few-shot learning [20], reinforcement learning [4], and transfer learning [46], and so on. For example, the authors of Reference [39] introduce a memory-augmented neural network to relearn the model when adequately incorporating new data without catastrophic interference. The authors of Reference [17] used a meta loss function to store the information of different environments. Lin et al. [29] propose a meta translation model to quickly adapt to a new domain translation task with limited training samples.

Recently, meta-learning is also utilized in personalized federated learning. The authors of Reference [21] and Reference [12] study different combinations of model-agnostic meta-learning–type methods with federated learning from an empirical point of view. By finding an initial global model, they make current or new clients can easily adapt to the local dataset by performing one or a few steps of gradient descent. The authors of Reference [23] treats meta-learning as the online learning of a sequence of losses that each upper bounds the regret on a single task. These methods can be considered as belonging to the local fine-tuning.

Different from all existing work, we design a meta-network for learning the collaborative relationship among clients from a fine-grained view and utilize the client's local data as supervised signals to train the network. Our method achieves adaptive selection of federations for clients and can be particularly effective when client data are heterogeneous.

3 PROBLEM FORMALIZATION

In a personalized federated learning system, there are m clients connected to a cloud server. Each client has the same type of base network $f(\cdot)$, and we use $X = \{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^{m\times d}$ to represent the set of local model parameters for all clients, where $x_i \in \mathbb{R}^d$ denotes the local model parameter of the client i. The cloud server collects the local model parameters uploaded by the clients and maintains a global model $x^g \in \mathbb{R}^d$ through the aggregation algorithm. By transmitting these model parameters, we can implement collaborative training among clients. For each client, we denote the objective function as $F(x_i) = \mathcal{L}(f(x_i); \mathcal{D}_i)$, where $\mathcal{D}_i$ denotes the local dataset of the client i and $\mathcal{L}$ is the loss function that measures the error between true values and those predicted by $f(x_i)$.

Due to the heterogeneous data in each client, we aim to analyze similarities among clients' local model parameters, and construct a fine-grained collaboration relationship to improve the performance of $f(x_i)$ beyond individual effort with the local dataset, while no local data are exposed to any other clients or cloud server. A HAM is proposed as the meta-network for personalized federated learning. FedHAM is deployed on each clients and aims to maintain a particular meta-model according to clients' characteristics. This meta-model can be used to generate model parameters for the base network in each client. Therefore, the optimization problem can be solved by

$$\min_{X\in\mathbb{R}^{m\times d},\Theta\in\mathbb{R}^d} \left\{ G(X,\Theta) := \sum_{i=1}^{m} F(x_i) + \lambda \sum_{i=1}^{m} F(\mathcal{H}_{\Theta^i}(x^g, X)) \right\}, \tag{1}$$

where $\mathcal{H}_{\Theta^i}(\cdot)$ denotes the meta-model of client i, Θ^i is the meta-model parameters, x^g denotes the global model parameters calculated by aggregation algorithm, X is the local model set for part of clients, and λ is the regularization coefficient.

Through Equation (1), x_i can be adjusted collectively by both the meta-network and the base network to obtain the optimal personalized model. Considering that the inherent complexity of most deep learning models usually makes it impossible to find closed solutions for X and Θ, we use gradient descent techniques and an alternative learning approach to solve the bi-optimization problem.

In the following, we introduce the network structure and implementation of HAM first. Then we describe the framework overview and training process of FedHAM, which is a **personalized federated learning (PFL)** framework for introducing HAM.

4 NETWORK STRUCTURE AND IMPLEMENTATION OF HAM

The HAM contains a hierarchical structure, which is defined by $\mathcal{H}_{\Theta^i}(x^g, X)$.

The structure of HAM is shown in Figure 1. It has two layers. The first layer automatically analyzes the similarities between the target client and the other clients. It aggregates the local model parameters of the other clients according to the weighted scores that are calculated by similarities. We introduce the aggregated results as an attention-enhanced model. In this way, we filter out model parameters of other clients that are not helpful to the target client. In the second layer, we aim to aggregate the attention-enhanced model and the global model parameters to achieve a tradeoff between clients' personalities and commonalities. In the following, we will give the detail of HAM design.

For a target client i, given the clients' local model parameter set X and the global model parameters x^g as input to train the HAM network, we define $g_1(\cdot)$ as an aggregation function of the first layer, which calculates an attention-enhanced model parameter x_i^a as the input of the second layer. Then, we define $g_2(\cdot)$ as the aggregation function of the second layer, which calculates the hybrid local model x_i^{hybrid}. The hybrid local model is the output of the HAM, and the process of

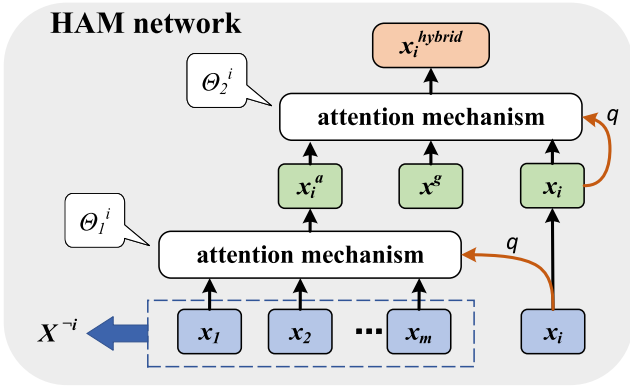


Fig. 1. The overall structure of HAM Network, which contains two layers. Θ_1^i and Θ_2^i are the learnable parameters in HAM.

its calculation is represented as

$$x_i^{hybrid} = \left\{\mathcal{H}_{\Theta^i}(x^g, X) := g_2(x_i, g_1(X; \Theta_1^i), x^g; \Theta_2^i)\right\}, \tag{2}$$

where $\{\Theta_1^i, \Theta_2^i\} \in \Theta^i$ denote all learnable parameter set in HAM. Equation (2) presents a formal computation of the two-layer structure of the HAM. In the following, we will explain the calculation of $g_1(\cdot)$ and $g_2(\cdot)$ separately.

The computation of $g_1(\cdot)$ in the first layer of HAM can be expressed as follows:

$$x_i^a = g_1(X; \Theta_1^i), \tag{3}$$

where Θ_1^i represents the learnable parameter set in the first layer of HAM. x_i^a is the output of the first layer of HAM, which represents the aggregation result of the other client's model parameters.

To implement model parameter aggregation, inspired by Reference [48], we automatically assign similarity scores to each input model parameters from other clients by applying an attention mechanism. Then, the model parameters are weighted summed by similarity scores to generate the attention-enhanced model, which reserves the knowledge that is beneficial for the target client. The essential idea of the attention mechanism is to calculate the relevance between a query q and a key matrix K, which can be written as

$$\mathbf{Att}(q, K) = \text{softmax}\left(\frac{qK^\top}{\sqrt{d}}\right), \tag{4}$$

where d is the scaling factor, which avoids overly large values of the inner product. In $g_1(\cdot)$, we split X into $X^{\neg i}$ and x_i, where $X^{\neg i} = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m\} \in \mathbb{R}^{(m-1)\times d}$. Referring to Equation (5), we treat x_i as a query matrix, $X^{\neg i}$ as the key–value matrix and calculate the attention scores according the following function:

$$g_1(X; \Theta_1^i) = g_1(x_i, X^{\neg i}; \Theta_1^i) = \mathbf{Att}(W_{q1}^i x_i, W_{k1}^i X^{\neg i}), \tag{5}$$

where $\{W_{q1}^i, W_{k1}^i\} \in \Theta_1^i$ are learnable parameters of query and key in the first layer in HAM, which are trained by gradient descent with the client's local dataset. Therefore, the calculation of the first layer can be rewritten as

$$x_i^a = \left\{g_1(X; \Theta_1^i) := \mathbf{Att}(W_{q1}^i x_i, W_{k1}^i X^{\neg i})\right\}. \tag{6}$$

Based on the learnable attention parameters Θ_1^i and the local model parameters set X, we can obtain the attention-enhanced model x_i^a.

According to our design, we assign high weights to those clients that have similar parameters with the target client to enhance their collaborations. Compared with the traditional federated learning approaches [6] that facilitate collaborations from the view of mathematical statistics, HAM analyzes clients' similarities from the parameter aspect.

In the second layer, given its local model parameters x_i, the attention-enhanced model x_i^a, and the global model parameters x^g as the input of $g_2(\cdot)$. The computation of $g_2(\cdot)$ can be expressed as follows:

$$x_i^{hybrid} = g_2(x_i, x_i^a, x^g; \Theta_2^i), \tag{7}$$

where Θ_2^i represents the learnable parameter set in the second layer of HAM. The model parameter x_i^{hybrid} aggregates the valuable information from the local model, attention-enhanced model, and global model.

Specifically, we use another attention mechanism method with the same structure as the first layer, and adopt x_i as the query matrix. The difference is that we concatenate x_i, x_i^a and x^g together as the key–value matrix. The computation can be expressed as follows:

$$\begin{aligned} &g_2(x_i, x_i^a, x^g; \Theta_2^i) = g_2(x_i, [x_i, x_i^a, x^g]; \Theta_2^i) = \mathbf{Att}(W_{q2}^i x_i, W_{k2}^i [x_i, x_i^a, x^g]), \\ &\text{where } \; x_i^a = g_1(X; \Theta_1^i), \end{aligned} \tag{8}$$

and $\{W_{q2}^i, W_{k2}^i\} \in \Theta_2^i$ are learnable parameters in the second layer of HAM. Based on Equation (8), the calculation of the second layer can be rewritten as

$$x_i^{hybrid} = \left\{ g_2(x_i, x_i^a, x^g; \Theta_2^i) := \mathbf{Att}(W_{q2}^i x_i, W_{k2}^i [x_i, g_1(X; \Theta_1^i), x^g]) \right\}. \tag{9}$$

As shown in Equations (2), (6), and (9), based on all learnable parameters Θ^i, local model parameter set X, and global model parameters x^g, we can obtain the final hybrid local model x_i^{hybrid} for client i in each communication round.

5 FEDHAM FRAMEWORK

5.1 Overview

Figure 2 shows the overall framework of FedHAM, which is constituted by a cloud server and several clients. Our proposed meta-learning model, HAM, is deployed on each client side. The main functions of the cloud server and clients in FedHAM are as follows.

The cloud *server* has three functions:

(1) Collecting each client's model parameters.
(2) Classifying local model parameters into k clusters by using clustering algorithm, such as k-means [24] or mean-shift [7] algorithm, and so on.
(3) Calculating the global model parameter x^g according to the aggregation algorithm.

When $k > 1$, X can be classified as $\{X_1, X_2, \ldots, X_k\}$. Each client receives only the corresponding cluster of the client's local model parameter set. The division of clusters does not affect the training process of FedHAM. Therefore, for simplicity, we use $k = 1$ for description in the following, which use X to represent the set of parameters in the HAM. In practice, the value of k is determined by the transmission capacity. We can set the k in FedHAM according to the practical situation. The larger k is, the smaller the set of local models each client receives. For each target client, it will only adopt the most similar $\frac{1}{k}$ model parameters that are from the other clients as the input X_i of the HAM network clustering of model parameters from other clients enables coarse-grained grouping of information provided by clients, which makes the training procedures can be

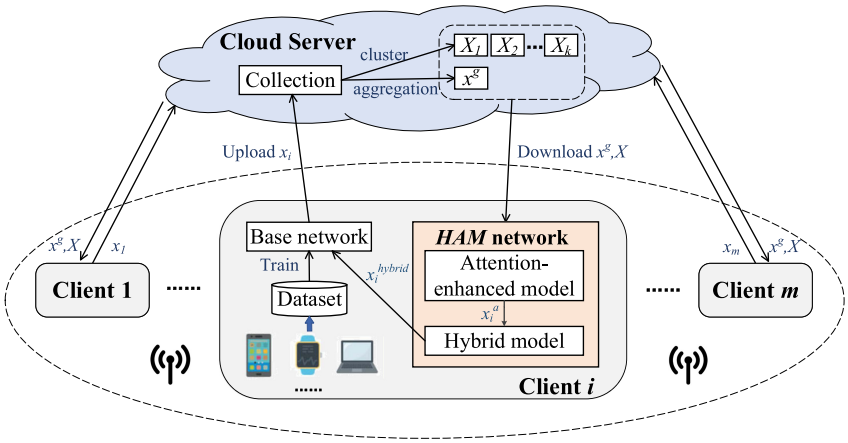


Fig. 2. The overall framework of FedHAM for personalized federated learning. Cloud server is used to implement model parameter collection, classification, and aggregation. Distributed deployment meta-models achieve relationships analyzing of clients and model fusion with parameter level.

performed without transferring all model parameters. Benefiting from the clustering process, not only the transmission overhead reduces but also the privacy leakages caused by the transmission are minimized. In the Section 6.7, we compare the model accuracy and training time when k is set at different values.

In addition, the global model here is a model with global information obtained by using existing algorithms, e.g., FedAvg [32] or its variants. By feeding the global model into HAM, we introduce global information as the complementary knowledge for personalized model generation, the global information also provides a soft starting for training because clients cannot perform refined modeling at the beginning.

The local *clients* have three functions:

(1) Collecting local datasets and pre-processing the data.
(2) Uploading and downloading model parameters from the server.
(3) Updating the local model parameters with the global model x^g or the hybrid local model x_i^{hybrid}.
(4) Training the base network and the HAM network with $\mathcal{D}_i$, x^g, and local model parameter set X, alternatively.

HAM is used to further increase the personalization performance based on the original local model. When the local model does not work well, it is also difficult for the HAM to achieve good training results. Since the introduction of HAM requires additional calculations, the HAM is not added to the training procedure at the beginning, and only the global model is used for cloud–client-side collaborative training. HAM joins the training procedure when the client's local model achieves stability (loss no longer decreases) or reaches a threshold of communication round. The stability of loss or the threshold value of communication round can be set manually. This training approach reduces the over-fitting of the model caused by the small amount of data in the local dataset and eliminates the wrong information brought by the irrelevant clients. Therefore, the inference accuracy of each client can be significantly higher than that of training the model using the local dataset alone or federated setting training.

In FedHAM, we utilize a meta-model for personalized model training, thus we need to co-train both the base model $f(\cdot)$ and the meta-model $\mathcal{H}_{\Theta^i}(\cdot)$. Due to their complicated dependencies on

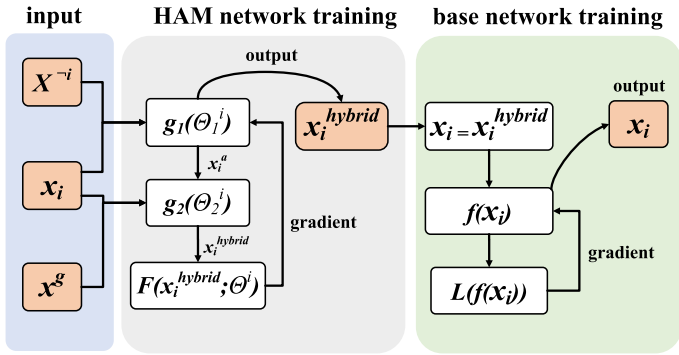


Fig. 3. The alternative training process for HAM network and base network in client.

each other, it is challenging to train them simultaneously. Therefore, we consider a bi-level optimization strategy [40] to learn our HAM. In this section, we discuss the case where the base model and meta-model are trained alternately at the same time. The training method of the base network is same as the common federated learning method and is not specifically described here.

The training process of FedHAM is shown in Figure 3. It contains two parts, which are HAM network training and base network training. In Figure 3, for simplicity, we ellipsis the function input, and use $g_1(\Theta_1^i)$ and $g_2(\Theta_2^i)$ to represent the two-layer function of the HAM network.

5.2 Training Process

First, in HAM network training step, to adjust the learnable parameter Θ^i in HAM, we need to embed it in the base network and train it using the client's local data. Specifically, the objective function of the base network ($F(x_i) = \mathcal{L}(f(x_i); \mathcal{D}_i)$) is used as the training target and trained using the gradient backpropagation method, which can be expressed as follows:

$$\Theta^{i*} = \arg\min_{\Theta^i} \{\mathcal{L}(f(\mathcal{H}_{\Theta^i}(x^g, X))); \mathcal{D}_i) := F(\mathcal{H}_{\Theta^i}(x^g, X))\}, \tag{10}$$

where Θ^{i*} denotes the optimal learnable parameters, $\mathcal{D}_i$ represents the local dataset of the client i, and $\mathcal{L}$ is the loss function that measures the error between true values and those predicted by $f(x_i)$. After training the HAM network using Equation (10), we fix the learnable parameters Θ^i and calculate the final hybrid model parameters x_i^{hybrid} for each client.

Then, we use x_i^{hybrid} as the initial value for the base network training and fine-tune it using the client's local data. The objective function of the base network training can be written as follows:

$$x_i^* = \arg\min_{x_i} \{\mathcal{L}(f(x_i); \mathcal{D}_i) := F(x_i)\}, \quad \text{where } x_i \text{ initialized with } \mathcal{H}_{\Theta^{i*}}(x^g, X), \tag{11}$$

and x_i^* denote the optimal local model parameters of client i. This result is used as the local model in this round for subsequent federated training.

In the above illustration of the training process, we consider only one client. By considering all clients, we can obtain the objective function $G(X, \Theta)$ as Equation (1). Based on $G(X, \Theta)$, the details of the server and client algorithm are shown in Algorithm 1 and 2. As shown in Algorithm 1, operations on the server side are similar to traditional federated learning, except that operations of clustering and transmitting X are increased. The improvements proposed in this article are mainly on the client side, as shown in Algorithm 2, and the training procedure for the client is as follows:

ALGORITHM 1: Model collection and aggregation (Procedure in the server).

Input: The set of local model parameters for all clients.
Initialize $t \leftarrow 0$, x^g as a init vector;
repeat
  Send $\{X_1, X_2, \ldots, X_k\}$ and x^g to each client;
  $t = t + 1$;
  for $i \in \{1, \ldots, m\}$ **do**
    Receive $x_i^{(t)} \in W$ from each client;
  Clustering of local model parameters as $\{X_1, X_2, \ldots, X_k\}$;
  Compute x^g according to the aggregation algorithm, such as $x^g \leftarrow \sum_{i=1}^{m} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} x_i$;
until *All node processes stop running*;
return $\{X_1, X_2, \ldots, X_k\}$ and x^g.

ALGORITHM 2: HAM network and base network training (Procedure in client i).

Output: Personalized model parameter x_i.
Initialize model parameter Θ^i and x_i randomly;
$t = 0$;
repeat
  Download X_i (which containing x_i) and x^g from the cloud server;
  $X \leftarrow X_i$;
  for *each local epoch* **do**
    if $t < p$ **then**
      for $\mathcal{D}_i \leftarrow$ *sample a mini-batch* **do**
        $x_i \leftarrow x_i - \alpha_1 \cdot \frac{\partial F(x_i)}{\partial x_i}$;
    else
      for $\mathcal{D}_i \leftarrow$ *sample a mini-batch* **do**
        $\Theta^i \leftarrow \Theta^i - \alpha_2 \cdot \frac{\partial G(X,\Theta)}{\partial \Theta^i}$;
      $x_i^{hybrid} = \mathcal{H}_{\Theta^{i*}}(x^g, X)$;
      $x_i \leftarrow x_i^{hybrid}$;
      for $\mathcal{D}_i \leftarrow$ *sample a mini-batch* **do**
        $x_i \leftarrow x_i - \alpha_1 \cdot \frac{\partial G(X,\Theta)}{\partial x_i}$;
  $t = t + 1$;
  Upload local model x_i to cloud server;
until model converge or $t > num$;
return x_i.

- Download X_i (containing x_i) and x^g from server. X_i is denoted by X in the following.
- For the first few communication rounds (which is set to p), we train the base network for each client i with $F(x_i)$ to obtain a stable local model x_i:

$$x_i \leftarrow x_i - \alpha_1 \frac{\partial F(x_i)}{\partial x_i}. \tag{12}$$

- After p rounds of training, we obtained robust models for each client. On this basis, the HAM network is added to alternate training with the base network.

First, train the HAM network by multiple gradient descent as follows:

$$\Theta^i \leftarrow \Theta^i - \alpha_2 \frac{\partial G(X, \Theta)}{\partial \Theta^i}. \tag{13}$$

Then, use the trained HAM network to infer the hybrid network parameters and use it to update the local model parameters:

$$x_i \leftarrow \mathcal{H}_{\Theta^{i*}}(x^g, X). \tag{14}$$

Finally, fine-tune the hybrid model using the base network and local data to get the final personalized model:

$$x_i \leftarrow x_i - \alpha_1 \frac{\partial G(X, \Theta)}{\partial x_i}, \tag{15}$$

where α_1 and α_2 are the learning rate.

The training will stop until the communication round is larger than the set value or the model converges. After finishing training, the client's local model is used as the final personalized model parameters.

In our method, most of the calculations are performed by the client. The cloud server maintains the clients' local and global models using collaborative learning from decentralized data. This training approach allows clients to have more useful information without using other clients' datasets. Also, it reduces the over-fitting of the model due to the small amount of data in the local dataset and eliminates the wrong information brought by the impurity client. Therefore, the accuracy of each client can be significantly higher than that of training the model using the local dataset alone.

6 EXPERIMENTS

In this section, we evaluate several state-of-the-art methods as baselines to compare the effect of FedHAM, which contains nine baseline methods. We give a brief introduction to them:

- **Local-Train** method trains each client's personalized model independently without communicating with the server.
- **FedAvg** [32] is the first proposed federated learning method, which utilizes weights average to enable all the clients to train a global model collaboratively.
- **FedSGD** [32] is a special case of FedAvg, which performs the global model aggregation after each iteration and thus equates to a weighted average of the gradients of the clients.
- **FedProx** [28] regularizes the distance between a global model and local models to prevent local models from deviating.
- **FedPer** [3] proposes a base + personalization method on the basis of federated learning, which only trains the base layer collaboratively.
- **FedHealth** [9] trains the cloud model on a public dataset in the server and then the clients train personalized models based on cloud model and local data.
- **FTL** [50] performs model aggregation through federated learning and then builds relatively personalized models by transfer learning.
- **FedBABU** [36] updates the body of the global model and the head of model are fine-tuned for personalization during the evaluation process.
- **FedAMP** [19] maintains a personalized cloud model on the cloud server for each client, and realizes the attentive message passing mechanism by attentively passing the personalized model of each client as a message to the personalized cloud models with similar model parameters.

Among them, FedSGD, FedAvg, and FedProx are the federated learning (FL for short) methods. FedPer, FedHealth, FTL, FedBABU, and FedAMP are PFL methods. For FedHealth, according to the method requirements, we adopt 10% of the training data in each client to form the public dataset at the server.

All the experiments are implemented in PyTorch 1.5.0 running on a 4 Tesla-P100 GPU cluster, with Intel Xeon E5-2620 CPU, 128 G memory, and Ubuntu 16.04.7. To simulate the transmission overhead of long-distance clients in the actual situation, we use the TCP protocol for reliable connection-oriented transmission.

To verify the generalizability of our FedHAM, we conducted experiments on datasets with different data distributions. The settings of dataset are described in the following Section 6.1.

6.1 Settings of Datasets

6.1.1 Datasets. We conduct experiments on two tasks: image classification and real-world urban air quality inference.

V-MNIST Dataset: This dataset is a collection of six benchmarks for variation mnist recognition, which are MNIST [26], MNIST with rotation, MNIST with noise background, MNIST with image background, MNIST with rotation and image background [45], and Fashion MNIST [51]. Both of them are 10-classification tasks. To simulate the personality data of each node, we perform data partitioning on these six variation MNIST datasets. Random samplings without replacement under the independently identical distribution are used. The whole dataset is partitioned into a certain number of shards, denoted as the number of clients. In the experiment, we divide each variation mnist dataset into three sub-datasets according to the proportion of the original amount of data. In total, 72,795 samples are divided into 18 sub-dataset, where one sub-dataset represents one client.

W&A of China Dataset: It is a real-world Weather and Air-quality dataset (W&A China), which is released by China National Environmental Monitoring Centre[1] and National Climate Data Center.[2] It was collected from January 1, 2017, to December 31, 2017, at an hourly interval from four municipalities in China (Beijing, Tianjin, Shanghai, and Chongqing). There are 230,044 records from 42 monitoring sites. Each W&A record comprises 13 feature elements: temperature, pressure, humidity, wind direction, wind speed, station ID, collection time, and the concentration of six pollutants. According to China's ambient air quality standard [33], air quality is divided into five levels according to the concentration of $PM_{2.5}$. We use the features of the past 48 hours to estimate the air quality in the next hour. The quantity and distribution of data across sites are heterogeneous and the site data are retained locally, making it suitable for personalized federated learning applications. Each site is considered as a client in our experiments.

6.1.2 Data Proportion. The amount of data on different clients is unbalanced in the actual scenario. For example, different monitoring stations have different sensing intervals, or sensors in specific locations are more likely to have data missing. We adopted two data division methods to explore the influence of different data distribution on training: unbalanced division and balance division. In the unbalanced division, for the V-MNIST dataset, we divide each variation mnist dataset into three sub-datasets according to the proportion of the original amount of data. For the W&A of China dataset, we use real data collected by each site in half a year as the node dataset. In the balance division, the amount of samples in each node is the same. To facilitate the distinction, we use (unb) to represent the unbalanced distribution and (b) to represent the unbalanced

[1]China National Environmental Monitoring Centre: http://www.cnemc.cn.
[2]National Climate Data Center: https://www.ncdc.noaa.gov.

Table 1. Statistics and Experiment Settings of Datasets

Dataset	Clients	Samples	Samples/client	
			mean	stdev
V-MNIST (unb)	18	72,795	4,044	1,876
V-MNIST (b)	18	82,800	4,600	0
W&A China (unb)	42	230,044	5,477	2,034
W&A China (b)	42	206,279	4,911	0

distribution. The statistics of datasets used in experiments are summarized in Table 1. We report the total number of edge nodes, the total number of samples, and the mean and deviation in the sizes of total sub-datasets on each node.

6.2 Base Network and Evaluation Metrics

Different base network structures can significantly affect the performance of personalized federated learning. To validate the generalizability of our method, we used two different base network structures (CNN and RNN) for different datasets and tasks.

For the V-MNIST dataset, a classical Alexnet [25] is employed as the underlying network used to classify the images. It is a CNN including five convolutional layers and three fully connected layers. The optimizer is mini-batch gradient descent, whose initial learning rate is set to 0.02. The loss function is Cross-Entropy Loss,[3] which is often used in multi-classification problems. We set the epoch per communication round as $E = 1$ and batch size in training as $B = 32$.

For the W&A of China dataset, a two-layer **Gated Recurrent Unit Network (GRU)** [10] is constructed with 128 hidden cells for each layer as the base network used for the time-series inference task. The result is then output through a fully connected layer and compared with the true value to obtain the prediction accuracy. The optimizer is mini-batch gradient descent, whose initial learning rate is set to 0.01. The loss function is MultiLabelSoftMargin Loss,[4] which is suitable for multi-object classification. We set the epoch per communication round as $E = 2$ and batch size in training as $B = 32$.

For V-MNIST and W&A China datasets, we take 150 and 175 as the communication rounds between cloud server and clients and select the best model based on the performance of the validation set. Then the mean testing results on all clients are taken as the final results.

Metrics: We use classification *accuracy* and *precision* as the metrics to evaluate the results. Accuracy is the ratio of all correct predictions (both positive and negative categories) to the total number of samples. Precision is the proportion of samples with correct predictions of positive categories to all samples with positive predictions. They can be expressed as follows:

$$\begin{aligned} accuracy &= \frac{TP + TN}{TP + FN + TN + FP}, \\ precision &= \frac{TP}{TP + FP}, \end{aligned} \tag{16}$$

where TP represents true positives, FN represents false negatives, TN represents true negatives, and FP represents false positives.

[3] https://pytorch.org/docs/master/generated/torch.nn.CrossEntro-pyLoss.html.
[4] https://pytorch.org/docs/master/generated/torch.nn.MultiLabel-SoftMarginLoss.html.

Table 2. Accuracy and Precision on the V-MNIST and the W&A China Datasets between Different Methods (All the Values in the Table Are Percentage Numbers with % Omitted)

Dataset		V-MNIST(unb)		V-MNIST(b)		W&A China(unb)		W&A China(b)	
Evaluation Metric		Accuracy	Precision	Accuracy	Precision	Accuracy	Precision	Accuracy	Precision
Separate	Local-Train	78.95	79.24	82.75	82.83	77.36	51.25	78.72	51.97
FL Methods	FedSGD	62.28	63.3	63.06	65.21	68.61	26.86	68.58	27.18
	FedAvg	81.76	78.1	81.83	81.81	72.57	46.09	72.08	52.42
	FedProx	81.85	78.64	82.01	82.15	73.42	48.31	73.23	54.11
PFL Methods	FedPer	82.23	78.42	83.12	82.14	79.09	60.16	78.34	61.57
	FedHealth	82.81	79.58	83.74	83.29	80.21	62.75	79.54	62.63*
	FTL	81.07	81.52	83.62	84.06	81.51	63.57	79.98	59.83
	FedBABU	81.53	82.22	83.91	84.42	81.89	63.64	80.13	61.29
	FedAMP	81.94*	82.68*	84.34*	84.82*	82.12*	64.32*	80.86*	61.31
	FedHAM	**85.58**	**85.71**	**87.73**	**87.98**	**84.57**	**66.91**	**84.26**	**66.32**
▲%		3.64	3.03	3.39	3.16	2.45	2.59	3.4	3.69

The best performance in each row is in bold, and the starred numbers represent the best baseline performance.

6.3 Comparison against Baselines

As shown in Table 2, we compare our method against several competitive baseline methods on two kinds of datasets with different base networks and data proportions.

6.3.1 Results on the Unbalance Data Setting. Local-Train is used as a baseline to represent the performance when there is no collaborative relationship between clients. It is only higher than FedSGD in the V-MNIST dataset while significantly outperforming the federated learning methods in the W&A China dataset. For the V-MNIST dataset, the local training data are not enough to support complex image recognition tasks. At this point, collaborative training among clients is essential. However, for the W&A China dataset, the data characteristics between sites in different cities are widely diverse, so it is difficult to perform well with the federated learning methods. It is worth noting that for the V-MNIST dataset, the precision is very close to the accuracy, but the W&A China dataset is relatively lower. Because the V-MNIST is a benchmark dataset in which the number of samples in different categories is almost evenly distributed, but W&A China is a real-world dataset. The time when the air quality is terrible is always less than when the air quality is good. Precision describes the proportion of the examples classified as positive that are positive. Usually, the precision of the model is lower in category imbalanced classification problems. This also explains why the two datasets have different effects on different baselines.

The federated learning methods FedSGD, FedAvg, and FedProx obtain lower performance than the personalized federated learning methods in both two datasets. Compared with our method, the best performing federated learning method (FedProx) is 3.73% and 11.15% less in accuracy in V-MNIST (unb) and W&A China (unb) datasets, respectively. This indicates the necessity of building personalized models for clients with heterogeneous data in collaborative training.

The personalized federated learning methods FedPer, FedHealth, FTL, and FedBABU are the local fine-tuning methods. Although the implementation methods are different, they achieve similar personalization performance. Among them, FedBABU performs the best, which is 4.05% and 2.68% lower in accuracy than our FedHAM in the V-MNIST (unb) and W&A China (unb) datasets, respectively. FedAMP is a multi-task method, which achieves the second-best performance among all methods by facilitating pairwise collaborations between clients without using a single global model. Compared with the second-best method, our method improves the accuracy by 3.64% and 2.45% and improves the precision by 3.03% and 2.59% in V-MNIST (unb) and W&A China (unb) datasets, respectively. In the distributed system, our HAM network takes model parameters of the

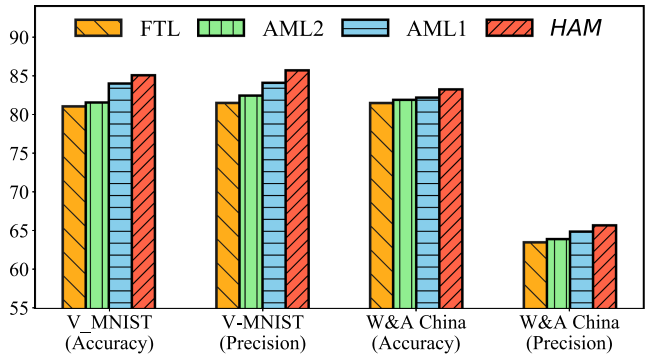


Fig. 4. Ablation study of *HAM* and three of its degenerate variants (FTL, AML1, and AML2) on two datasets with unbalance data setting.

client as features, and learns a meta-model to analyze similarities among various clients automatically for better performance.

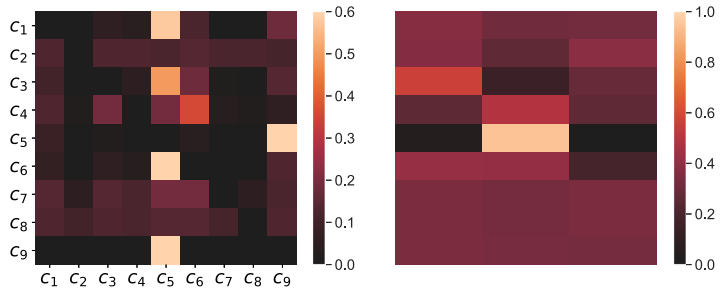
6.3.2 Results on the Balance Data Setting. In Table 2, V-MNIST(b) and W&A China(b) datasets show the experimental results of each method when balancing the data distribution. It can be seen that the balanced data setting results in a significant improvement of the precision in the federated learning method. On the V-MNIST dataset, compared with the unbalanced distribution, the precision of FedAvg and FedProx methods increased by 3.71% and 3.51%. On the W&A dataset, the precision rate increased by 6.33% and 5.8%, respectively. This indicates that the unbalanced data setting has a negative impact on the federated learning method using the global model. This is because the use of non-iid for data training brings significant instability to the global model aggregation [54].

At the same time, for the existing PFL methods, the distribution of the dataset will also affect the experimental effect. Compare to PFL methods, our FedHAM improves the accuracy by 3.39–4.61% and 3.4–5.92% in V-MNIST (b) and W&A China (b) datasets, respectively. For the best-performing PFL method, FedAMP, the accuracy difference brought by the data distribution in the two datasets reached 2.40% and 1.26%, respectively. In contrast, the difference in the accuracy of our HAM method in different data distributions is only 2.15% and 0.31%. It can be seen that, compared with the existing methods, our FedHAM can significantly reduce the negative impact of different data distributions.

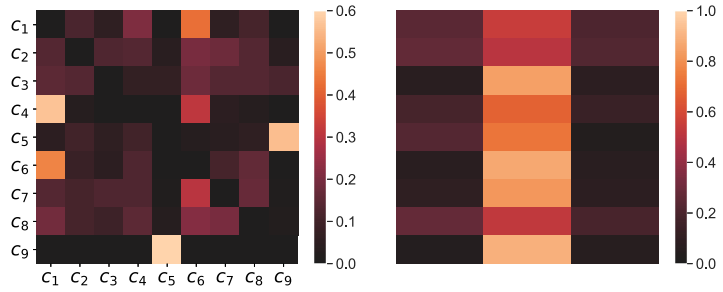
6.4 Ablation Study

We use HAM to fuse multi-model parameters in personalized federated learning. In this section, we conduct ablation study experiments by removing each modules of HAM to analyze their influences. For clearly discussion, we make some degradation of HAM. Three sub-variant methods are considered. (1) FTL: we do not add HAM to the client and only use a global model to promote the personalized model training; (2) AML1: we only add the first layer of HAM to generate the attention-enhance model and use this model to promote the personalized model training; and (3) AML2: we only add the second layer of HAM to generate hybrid local model and use this model to promote the personalized model training. Note that only the local and global models are used as network inputs since there is no attention-enhanced model in AML2.

The accuracy and precision results of HAM and three sub-variant methods on two datasets are shown in Figure 4. It can be seen that our method is better than all ablation methods, which shows



(a) communication round = 2



(b) communication round = 50

Fig. 5. The visualization of the attention scores computed by the first (left) and second (right) layer of HAM in the W&A China dataset under nine clients from two cities with different communication rounds. The color of the grid represents the attention score. The lighter the color the stronger the correlation. It can be seen that as the number of communication rounds increases, the inter-client correlation changes and the proportion of x_i^a increases significantly.

that HAM can improve performance. Specifically, our method is 3.26% and 3.45% better than FTL in average accuracy and precision, because it is difficult to model the client personalized using only the global model. The performance of AML1 and AML2 are better than FTL and weaker than HAM in both accuracy and precision. The addition of the attention-enhanced model has brought an average of 1.04% and 2.77% improvement in accuracy and precision. Meanwhile, we found that AML1 brings more benefits than AML2 because the first layer absorbs valuable information from other clients. Although the benefits of the second layer are slightly lower than the first layer, it still brings us further performance improvement after fusing these model parameters.

6.5 Visualization of HAM Network

In this section, we visualize attention scores over clients to further understand how HAM enhances collaborations between the clients whose model parameters are similar. Specifically, we randomly select model parameters of nine clients from the W&A China dataset, denoted as $c_1, c_2, \ldots, c_9$, where c_5 and c_9 are clients located in Shanghai, the rest clients are in Beijing.

Figure 5 shows the visualized attention scores of nine clients in different communication rounds. Figure 5(a) shows the distributions of attention scores of nine clients at the second communication round, and Figure 5(b) displays the distributions of attention scores of nine clients at the 50th communication round. In Figure 5(a), we observe that the weight of each client in the first layer is biased toward the average or a particular client, and the three models (x_i, x_i^a, and x^g) weights

Table 3. Comparison Experiments of Communication Rounds and Communication Overhead Required by Different Baseline Methods to Achieve the Same Benchmark Accuracy

	Accuracy	Communication Rounds	Communication Overhead
FedPer	79.5	200	180×
FedHealth	79.4	200	260×
FTL	80.0	163	163×
FedBABU	80.0	169	256×
FedAMP	80.0	137	415×
FedHAM	80.0	106	471×

in the second layer are average except for c_5. This indicates that both the clients' local and global models are not well trained at the early stage of training, and the meta-model cannot generate appropriate model parameters at the beginning.

In Figure 5(b), we can see that both c_5 and c_9 assign the highest attention scores to each other, because the dataset collected by clients of c_5 and c_9 are from Beijing and the model trained from the similar dataset have higher similarity. Attention scores between (c_5 and c_9) and other clients's model parameters are much smaller since models trained from different distributions of dataset are distinguished and will provide less information for generate the personalized model parameters. Meanwhile, these experimental results shows that at the early stage of training, the local model is not sufficiently trained, and the addition of HAM does not bring good results. However, at the later stage of training, HAM can be well implemented to group similar clients and enhance their collaborations by assigning high scores. We also find that in the second layer, the global model, local model, and the attention model can all benefit from personalized federated learning according to their visualized attention scores, while our constructed attention-enhance model plays the most significant role in most cases.

6.6 Comparison of Communication Overhead

To investigate the impact of introducing HAM in FedHAM on the communication overhead, we compare the total number of communication rounds and the total communication overhead that different personalized federation learning methods need to spend to achieve the same accuracy in this section. The experiments use the W&A China dataset with a two-layer GRU as the base network. For each method, we selected an average accuracy of 80% as the benchmark, with FedPer and FedHealth failing to reach the benchmark accuracy even with the maximum number of communication rounds. Therefore, we recorded the highest achievable accuracy for these methods and the number of communication rounds required to achieve the accuracy for different methods. We defined the communication overhead of transmitting a base network once as 1.0× and calculated the total communication overhead based on the number of communication rounds and the communication overhead per round for different methods.

The experimental results are shown in Table 3. It can be seen that FTL has the lowest communication overhead because it does not need to have additional computations and only needs to fine-tune the global model at each communication round. In contrast, methods such as FedBABU, FedAMP and FedHAM bring higher communication overhead while achieving higher accuracy due to the need to utilize additional modules for personalization. In particular, our FedHAM, despite a 35% decrease in the number of communication rounds, still introduces more than twice the total communication overhead compared to FTL. To alleviate this situation, we further test the clustering algorithm used for coarse classification in the server in FedHAM in Section 6.7, so that

Table 4. Comparison Experiments of Different Clustering Algorithms in FedHAM on Accuracy and Training Time with Baseline Methods

	Local-Train	FedAvg	FedProx	FTL	FedBABU	FedAMP	FedHAM (clustering)				
							k = 1	k = 3	k = 5	k = 10	mean-shift
Accuracy	76.54	70.01	71.69	82.72	82.14	82.83	84.29	84.60	**84.73**	83.95	83.26
ToA@70	18.2	60.4	73.7	55.4	72.5	128.6	349.7	231.5	189.3	**164.8**	173.7
ToA@80	—	—	—	104.6	156.2	239.3	526.7	375.4	326.8	**292.4**	317.1

ToA@: the time (in minutes) required to arrive at a testing classification accuracy (the smaller the better).

clients can get a comprehensive understanding of the application scenarios for which FedHAM is suitable.

6.7 Effect of Clustering Algorithm on Model Accuracy and Training Time

The introduction of HAM increases the communication overhead between clients and the server, and to reduce this impact, we add clustering algorithms to the server. When the number of clients is large, the server coarsely classifies them, and each client downloads only the corresponding clusters with its own local model parameters. To verify the effect of different clustering algorithms or values of k on model accuracy and training time, we designed experiments as follows.

Dataset: We used the data collected from air quality monitoring sites in the W&A China dataset as client data. Based on the original four Chinese municipalities, we added other 27 monitoring sites in Chinese provincial capitals as the experimental data. We selected 304 sites distributed in 31 Chinese municipalities or provincial capitals for the experiment, containing a total of 1.59 million samples.

Comparison methods: We compared different clustering algorithms, e.g., the k-means method (in the case that k takes different values) and the mean-shift method, on FedHAM. Also, we compared the model accuracy and training time in different federated learning or personalized federated learning methods. We used the *time of arrival at a desired accuracy (ToA@)* as defined in Reference [35] to evaluate the training time in our experiments. By observing the variation of accuracy with time, we record the time when the set value of model accuracy is reached. For example, *ToA*@70 represents the time taken to reach 70% accuracy for the first time, and smaller values represent faster training.

As shown in Table 4, compared with Local-Train, federated learning methods and personalized federated learning methods, our FedHAM is at least 8.19%, 13.04%, and 1.9% higher in accuracy. The mean-shift method divides the sites into 13 clusters, which can be considered as $k = 13$ in this case. We found that the difference in accuracy of FedHAM on different clustering methods is not significant. As k increases, the accuracy of FedHAM shows a trend of increasing and then decreasing. This is because when the value of k is small, other sites that are beneficial to the target site are more likely to appear in the same cluster, while irrelevant sites may be divided into other clusters. However, when the value of k is large, the set of parameters received for the target site decreases, and the number of beneficial other sites in it decreases significantly, affecting the accuracy of personalized modeling.

In terms of the *ToA*@ metric, we find that the total training time of FedHAM decreases as the value of k increases. This is due to the fact that the set of model parameters downloaded by the site becomes smaller and the transmission time decreases. It can be seen that when $k = 1$, the training time of FedHAM is longer compared to other baseline methods. However, when $k = 10$, the training time of FedHAM is only 53.4 minutes slower than FedAMP, with an average accuracy of 1.12% higher. In addition, although the mean-shift method divides the set of model parameters into 13 clusters, its training time is greater than that of the k-means method when

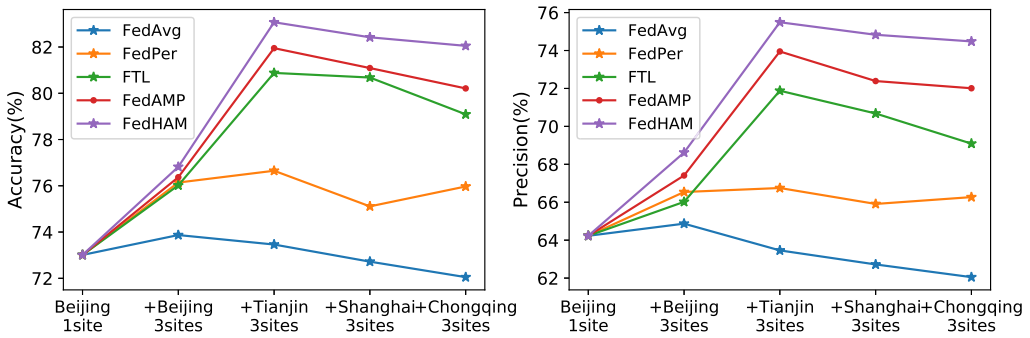


Fig. 6. A case study in the W&A China dataset, which explores the effect of different methods performance when changing the number of clients in different cities for collaborative training.

$k = 10$. This is because the computation time of the mean-shift algorithm is greater than k-means algorithm. Therefore, in practical situations, it is possible to achieve faster FedHAM training times by specifying the value of k in the k-means algorithm.

6.8 Case Study

To obtain a better understanding of why FedHAM performs better than other models, in this section, we conduct the case study to compare FedHAM and the other models qualitatively. We conduct experiments on W&A China dataset, and we select 42 clients that belong to four cities (Beijing, Tianjin, Shanghai, and Chongqing) in China. We randomly reserve three clients for each city to reduce the impact of the number of sites on the results. We use a site in Beijing as the observed client during the training process and add Tianjin, Shanghai, and Chongqing sites in turn. By adjusting the number of participating sites and cities in the collaborative training, we compare the impact of different methods on the accuracy of the observed client personalized model to test whether the model performs well in the face of data heterogeneity problems. Since the FedHealth method requires the construction of the source domain dataset on the server, we do not use it as a comparison method here. In Figure 6, we compare our FedHAM with the baseline methods, and evaluate them with the accuracy and the precision.

As shown in Figure 6, when only observed client is available, there is no need for collaborative learning in different methods, so we record the Local-train result of the observed client (site) as 73.01% and 64.2%, in accuracy and precision, respectively. After that we add the sites of different cities in turn. The performances of all the methods improves when three cities in Beijing are introduced in the collaborative training. We observe that our FedHAM outperforms the best baseline in terms of both accuracy and precision by 0.81% and 1.12% compared to the best baseline. With the addition of the three sites in Tianjin, there are 7 sites involved in collaborative training. The the accuracy and precision of the observed client in FedHAM increased by 6.25% and 6.8%, which has a greater increase than other methods. When the sites of Shanghai and Chongqing are added in our experiments, the performances of all methods are decreased. However, the accuracy and precision of our FedHAM still outperforms the best baseline by 3.13% and 2.94% when three sites of Shanghai are added and the performances of the observation client in all methods are degraded. This is easy to understand, as Beijing and Tianjin are geographically close. The two cities have similar trends of air quality, and their site data can facilitate training So they perform better than the local training approach. However, when southern cities such as Shanghai and Chongqing had a completely different data distribution than northern cities, their performance declined. From the experimental results, we find that the personalized federated learning methods (FedPer, FTL,

FedAMP, and FedHAM) can achieve better performance in the case of complex sites compared to the federated learning methods (FedAvg). Further more, we find that FedAMP and our FedHAM suffer the least negative impact, and FedHAM shows the best performance. It demonstrates that by assigning a meta-learning approach to learn their similarities automatically, our FedHAM is more robust and effective when facing a serious data heterogeneous problem between clients.

7 DISCUSSION: HOW TO ENSURE DATA PRIVACY

The FedHAM proposed in this article needs to deploy a HAM network to clients and use the local model parameters of each client as input for personalized model generation. It does not need to disclose the client's local data, which can guarantee data privacy to a certain extent. However, the method requires exposing the client's local model parameter, so there is a potential risk that an attacker can infer the client's local data through model parameter analysis. To ensure data security, this section combines FedHAM with privacy protection techniques by investigating existing data privacy protection techniques to provide further security assurance for each client.

Privacy leakage and defense of machine learning is a dynamic process. In terms of technical means, there are two typical attacks that FedHAM may face, which are model inversion attack and membership inference attack [34].

(1) **Model Inversion Attack:** It mainly refers to attackers extracting information related to training data from model prediction results [14]. For example, the base network is a neural network for face recognition, and the attacker gets the network parameters by hacking or pretending to be a client. The input image of a face can output the predicted person's name and the corresponding confidence level. The attacker can construct a random image with the prediction confidence of a person (e.g., Rita) in the training data as the target, and use methods such as gradient descent to repeatedly correct the image based on the network prediction results to obtain an image with higher confidence of Rita's prediction. This process indirectly obtains the private picture information about Rita in the training set. In particular, this attack is particularly effective when combined with generative adversarial networks [16, 53].
(2) **Membership Inference Attack:** It means that an attacker learns whether feature data are included in the training set of the model from the prediction results by accessing the model prediction API [42]. For example, in a model for medical inference, an attacker who knows that a person's medical records are involved in training a model for a specific disease may infer that the person has that disease. The attack process is as follows: First, the attacker obtains classification confidence based on the prediction API and uses it to obtain a high confidence dataset $\mathcal{D}$. Then, the attacker uses the dataset $\mathcal{D}$ to train a shadow model to observe whether data in $\mathcal{D}$ can cause a change in the prediction classification confidence. Finally, the attacker inputs user data into the base model and uses the API return results as input to the attack model to confirm whether the user appears in the training dataset. Compared with the model inversion attack, this method only needs to get the confidence of the predicted classification. It does not need to know the model structure, training method, model parameters, training set data distribution, and so on [38, 44].

FedHAM is suitable for application scenarios with strict privacy requirements, so it is necessary to take effective precautions to reduce the risk caused by model privacy attacks. The defenses against the above two attacks can be divided into the following three categories:

(1) **Confidence Fuzzy Processing:** Attack methods usually rely on the confidence value of the model output for each predicted classification. If the confidence values are reasonably fuzzified, such as rounding [47], multiple result aggregation [2], and homomorphic

encryption [52], then the accuracy of the predictive classification information can be guaranteed while reducing the accuracy of the exposed confidence level. It is difficult for the attacker to pounce on the detailed changes in the confidence level during the trial, thus increasing the difficulty of convergence of the attack.

(2) **Differential Privacy Protection:** It can add random noise to sensitive information during model training or in the final model parameters, making it impossible for an attacker to detect the effect of changes in the original training data on the model output [1]. Differential privacy techniques increase the difficulty of attacks, such as model extraction and training set detection, which are widely used in federated learning [18, 49].

(3) **Ensemble Learning Method:** The membership inference attack technique relies on the principle that for inputs with specific characteristics, the predicted output of the model can be easily distinguished from the predicted results of other inputs. Based on the above characteristics, we can segment the training data and train multiple models separately using subsets of the dataset. Then, using the ensemble learning methods to vote on the output results of multiple models. So that the final prediction results are shielded from the effects of model overfitting phenomenon, as a way to prevent membership inference attacks [30, 37].

The above three methods to prevent attacks can be integrated into FedHAM to deal with the privacy leakage problem caused by possible attacks. Moreover, the problem of client privacy leakage by similarity information among clients is usually possible. However, in FedHAM, we can adjust the number of client model parameters in the cluster by setting the value of k. When the value of k is small, the number of clients contained in a cluster is larger and the similarity of clients in a cluster is weak. It is difficult for the attacker to obtain detailed clients' similarity information. At the same time, for each client, they do not obtain information about the identity of other clients. Even if they find a client's model parameters are very similar to them, they cannot distinguish exactly which client it is. By the above two manners, the risk of clients' privacy leakage can be greatly reduced.

8 CONCLUSION AND FUTURE WORK

In this article, we formalize the personalized federated learning problem into a meta-learning task and introduce a novel HAM to solve the local model personalization problem. By treating model parameters as features, we design a two-layer network structure and utilize the attention mechanism to learn the similarities among the different clients automatically. An alternative learning approach is further applied to enhance the stability and flexibility of training. According to this design, HAM can reasonably achieve a tradeoff between clients' personalities and commonality. Extensive experiments based on two datasets are performed to prove that our method outperforms state-of-the-art baselines under different evaluation metrics.

Since transferring the set of model parameters increases the additional communication overhead. In the future, we will continue to investigate model compression algorithms for the FedHAM framework to reduce the communication problems between the server and the client.

ACKNOWLEDGMENTS

In addition, thanks to Dr. Li Shen of JD Explore Academy for his contribution to the problem formalization, which has a positive effect in improving the quality of the article.

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 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Mohammad Al-Rubaie and J. Morris Chang. 2016. Reconstruction attacks against mobile-based continuous authentication systems in the cloud. *IEEE Trans. Inf. Forens. Secur.* 11, 12 (2016), 2648–2663.
[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. arXiv: 1912.00818. Retrieved from https://arxiv.org/abs/1912.00818.
[4] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. 2017. Neural optimizer search with reinforcement learning. (unpublished).
[5] Mahantesh N. Birje and Savita S. Hanji. 2020. Internet of things based distributed healthcare systems: A review. *J. Data Inf. Manage.* 2, 3 (2020), 149–165.
[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H. Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. (unpublished).
[7] Miguel A. Carreira-Perpinan. 2007. Gaussian mean-shift is an EM algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 5 (2007), 767–776.
[8] Hong-You Chen and Wei-Lun Chao. 2021. On bridging generic and personalized federated learning. arXiv:2107.00778. Retrieved from https://arxiv.org/abs/2107.00778.
[9] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. 2020. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intell. Syst.* (2020).
[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. (unpublished).
[11] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*. PMLR, 2089–2099.
[12] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning: A meta-learning approach. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems (NeurIPS'20)*.
[13] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. 2020. Pmf: A privacy-preserving human mobility prediction framework via federated learning. *Proc. ACM Interact. Mob. Wear. Ubiq. Technol.* 4, 1 (2020), 1–21.
[14] 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.
[15] Filip Hanzely and Peter Richtárik. 2020. Federated learning of a mixture of global and local models. (unpublished).
[16] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: Information leakage from collaborative deep learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 603–618.
[17] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. 2018. Evolved policy gradients. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems (NeurIPS'18)*. 5400–5409.
[18] Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, and Yanmin Gong. 2020. Personalized federated learning with differential privacy. *IEEE IoT J.* 7, 10 (2020), 9530–9539.
[19] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. 2020. Personalized cross-silo federated learning on Non-IID data. In *Proceedings of the AAAI Annual Conference on Artificial Intelligence (AAAI'20)* (2020).
[20] Muhammad Abdullah Jamal and Guo-Jun Qi. 2019. Task agnostic meta-learning for few-shot learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 11719–11727.
[21] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. 2019. Improving federated learning personalization via model agnostic meta learning. (unpublished).
[22] 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. (unpublished).
[23] Mikhail Khodak, Maria-Florina F. Balcan, and Ameet S. Talwalkar. 2019. Adaptive gradient-based meta-learning methods. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems (NeurIPS'19)*. 5917–5928.
[24] K. Krishna and M. Narasimha Murty. 1999. Genetic K-means algorithm. *IEEE Trans. Syst. Man Cybernet. B (Cybernet.)* 29, 3 (1999), 433–439.
[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.
[26] Yann LeCun. 1998. The MNIST Database of Handwritten Digits. Retrieved from http://yann.lecun.com/exdb/mnist/.
[27] Ming-Chang Lee and Jia-Chun Lin. 2020. DALC: Distributed automatic LSTM customization for fine-grained traffic speed prediction. arXiv:2001.09821. Retrieved from https://arxiv.org/abs/2001.09821.

[28] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2018. Federated optimization in heterogeneous networks. (unpublished).
[29] Jianxin Lin, Yijun Wang, Zhibo Chen, and Tianyu He. 2020. Learning to transfer: Unsupervised domain translation via meta-learning. In *Proceedings of the AAAI Annual Conference on Artificial Intelligence (AAAI'20)*. 11507–11514.
[30] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. arXiv:2006.07242. Retrieved from https://arxiv.org/abs/2006.07242.
[31] Xiaofeng Lu, Yuying Liao, Chao Liu, Pietro Lio, and Pan Hui. 2021. Heterogeneous model fusion federated learning mechanism based on model mapping. *IEEE IoT J.* (2021).
[32] 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*. 1273–1282.
[33] AQSIQ MEP. 2012. *GB 3095-2012 Ambient Air Quality Standards*. China Environmental Science Press, Beijing (2012).
[34] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Fut. Gener. Comput. Syst.* 115 (2021), 619–640.
[35] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *Proceedings of the IEEE International Conference on Communications (ICC'19)*. IEEE, 1–7.
[36] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. 2021. Fedbabu: Towards enhanced representation for federated image classification. arXiv:2106.06042. Retrieved from https://arxiv.org/abs/2106.06042.
[37] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. arXiv:1610.05755. Retrieved from https://arxiv.org/abs/1610.05755.
[38] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. arXiv:1806.01246. Retrieved from https://arxiv.org/abs/s1806.01246.
[39] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of the International Conference on Machine Learning (ICML'16)*. 1842–1850.
[40] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. 2019. Truncated back-propagation for bilevel optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'19)*. PMLR, 1723–1732.
[41] Tao Shen, Jie Zhang, Xinkang Jia, Fengda Zhang, Gang Huang, Pan Zhou, Fei Wu, and Chao Wu. 2020. Federated mutual learning. (unpublished).
[42] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'17)*. IEEE, 3–18.
[43] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. 2017. Federated multi-task learning. In *Advances in Neural Information Processing Systems*. 4424–4434.
[44] Liwei Song, Reza Shokri, and Prateek Mittal. 2019. Membership inference attacks against adversarially robust deep learning models. In *Proceedings of the IEEE Security and Privacy Workshops (SPW'19)*. IEEE, 50–56.
[45] Kai Sun, Jiangshe Zhang, Hongwei Yong, and Junmin Liu. 2019. FPCANet: Fisher discrimination for principal component analysis network. *Knowl.-Bas. Syst.* 166 (2019), 108–117.
[46] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. 2019. Meta-transfer learning for few-shot learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 403–412.
[47] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security'16)*. 601–618.
[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems (NeurIPS'17)*. 5998–6008.
[49] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H. Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forens. Secur.* 15 (2020), 3454–3469.
[50] Qiong Wu, Kaiwen He, and Xu Chen. 2020. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Comput. Graph. Appl.* (2020).
[51] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. (unpublished).
[52] Peichen Xie, Bingzhe Wu, and Guangyu Sun. 2019. Bayhenn: Combining bayesian deep learning and homomorphic encryption for secure dnn inference. arXiv:1906.00639. Retrieved from https://arxiv.org/abs/1906.00639.

[53] Jiale Zhang, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. 2019. Poisoning attack in federated learning using generative adversarial nets. In *Proceedings of the 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE'19)*. IEEE, 374–380.

[54] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. 2020. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. arXiv:2005.11418. Retrieved from https://arxiv.org/abs/2005.11418.

Received 4 June 2022; revised 15 March 2023; accepted 27 March 2023