FedTP: Federated Learning by Transformer Personalization

Hongxia Li, Zhongyi Cai, Jingya Wang, Jiangnan Tang, Weiping Ding, Senior Member, IEEE, Chin-Teng Lin, Fellow, IEEE, and Ye Shi, Member, IEEE

Abstract—Federated learning is an emerging learning paradigm where multiple clients collaboratively train a machine learning model in a privacy-preserving manner. Personalized federated learning extends this paradigm to overcome heterogeneity across clients by learning personalized models. Recently, there have been some initial attempts to apply transformers to federated learning. However, the impacts of federated learning algorithms on self-attention have not yet been studied. In this article, we investigate this relationship and reveal that federated averaging (FedAvg) algorithms actually have a negative impact on self-attention in cases of data heterogeneity, which limits the capabilities of the transformer model in federated learning settings. To address this issue, we propose FedTP, a novel transformer-based federated learning framework that learns personalized self-attention for each client while aggregating the other parameters among the clients. Instead of using a vanilla personalization mechanism that maintains personalized self-attention layers of each client locally, we develop a learn-to-personalize mechanism to further encourage the cooperation among clients and to increase the scalability and generalization of FedTP. Specifically, we achieve this by learning a hypernetwork on the server that outputs the personalized projection matrices of self-attention layers to generate clientwise queries, keys, and values. Furthermore, we present the generalization bound for FedTP with the learn-to-personalize mechanism yields state-of-the-art performance in the non-IID scenarios. Our code is available online https://github.com/zhyczy/FedTP.

Index Terms—Hypernetworks, learn-to-personalize, personalized federated learning, self-attention, transformer.

I. INTRODUCTION

FEDERATED learning [1] is a framework that learns a shared global model from multiple disjointed clients without sharing their own data. In federated learning, each client trains a model using its own local data and only sends model updates back to the server. In this way, federated learning overcomes a range of problems with both data privacy and communication overheads. However, in situations where there is data heterogeneity and system heterogeneity across the clients, learning a single global model may fail. Accordingly, personalized federated learning has emerged as an extension to federated learning to overcome these challenges. The paradigm works by learning personalized models instead of a single global model, while still benefiting from joint training.

Data heterogeneity is a common problem in the real-world, since the training data collected from different clients vary heavily both in size and distribution [2], [3], [4], [5], [6]. Numerous methods have been developed for federated learning to tackle the challenge of non-IID data distribution across clients. Yet most federated learning frameworks are based on convolutional neural networks (CNNs), which generally focus on high-frequency local patterns that might be particularly sensitive to data heterogeneity [7]. Transformers [8], however, use self-attention to learn the global interactions between inputs [9] and, therefore, they tend to be more robust to distribution shifts and data heterogeneity [10]. Motivated by this, a very recent work [11] proposed the idea of using transformers as a federated network architecture coupled with the basic federated averaging (FedAvg) algorithm [1]. Although the study showed some very promising experimental results, the impacts that federated learning algorithms may have on self-attention have not yet been studied. It is our fear that such algorithms may limit the capabilities of transformers in federated learning. Given the promise of transformer-based federated learning, this is a topic worthy of further investigation.

To sum up, the main issues of current personalized federated learning are as follows.

1) Many existing CNN-based federated models might be particularly sensitive to data heterogeneity since CNNs generally focus on high-frequency local patterns.
these methods are shown in Fig. 1. We can see that both local-
The original images and the corresponding attention maps of
the Attention Rollout method [14] to produce attention maps.
different classes in CIFAR-100 [13] with five clients and using
The experiment itself involved sampling a set of images from
uses FedAvg to aggregate the other parameters in the server.
ViT in each client locally; 2) FedAvg-T, which applies the
models [12], these being: 1) Local-T, which trains a unique
the attention maps of three different vision transformer (ViT)
and to study the effects of federated learning methods on
increase in client numbers. Furthermore, the generalization
layers are not scalable as they increase linearly with the
may be suboptimal. Moreover, the personalized self-attention
client-specific self-attention layers by local training. However,
model, the above vanilla personalization mechanism can depict
aggregating the self-attention of clients with heterogeneous
To reveal that FedAvg may have negative impacts on
self-attention where data heterogeneity is present. Based
on this, we propose a novel transformer-based federated
learning framework, namely, FedTP, which learns per-
sonalized self-attention for each client.
2) We propose a learn-to-personalize mechanism to better
exploit clients’ cooperations in the personalized layers
and improve the scalability and generalization of FedTP.
In addition, we derive the generalization bounds for
FedTP with the learn-to-personalize mechanism.
3) We conducted extensive experiments on three bench-
mark datasets under different non-IID data settings. The experimental results demonstrate that FedTP yields
state-of-the-art performance over a wide range of per-
sonalized federated learning benchmark methods on both
image and language tasks.
The rest of this article is organized as follows. Section II
briefly reviews the related work of personalized federated
learning, transformers, and hypernetworks. Section III presents
the formulation of transformer-based personalized federated
learning; the details of two types of personalization, includ-
ing vanilla personalization and learn-to-personalize; and the
update of model parameters. The experimental results and
discussions are provided in Section IV. Section IV analyzes
the limitations of FedTP, and Section V concludes the whole
article. The proof of Theorem 1 in Section III is given in the
Appendix.
II. RELATED WORK
A. Personalized Federated Learning

Multiple approaches to personalized federated learning have
been proposed as a way of overcoming heterogeneity across
clients [15], [16], [17]. Currently, these methods can be divided
into two main categories: global model personalization and
learning personalized models [18]. Global model personal-
ization aims to improve the performance of a single shared
global model given heterogeneous data. An intuitive method
is fine-tuning the global model on the clients’ local datasets
to produce personalized parameters [19], [20], [21], [22].

2) Existing transformer-based federated learning methods
do not investigate the effects of the federated aggregation
operations on self-attention.
3) There is a lack of a personalized federated learning
framework that is more appropriate for transformer
structures.

Recently, it has been demonstrated that the self-attention
layers in transformers play a more important role than the other
layers [10]. Inspired by this, we devised a simple experiment
to explore the contribution of self-attention to federated learning
and to study the effects of federated learning methods on
the self-attention mechanisms. In our experiment, we compared
the attention maps of three different vision transformer (ViT)
models [12], these being: 1) Local-T, which trains a unique
ViT in each client locally; 2) FedAvg-T, which applies the
FedAvg algorithm to train a global ViT; and 3) Vanilla
Personalized-T, which retains the self-attention locally and
uses FedAvg to aggregate the other parameters in the server.
The experiment itself involved sampling a set of images from
different classes in CIFAR-100 [13] with five clients and using
the Attention Rollout method [14] to produce attention maps.
The original images and the corresponding attention maps of
these methods are shown in Fig. 1. We can see that both Local-
T and Vanilla Personalized-T can discover critical information
in the images (the red area in the images), but FedAvg-T failed
to generate a meaningful attention map. This indicates that
aggregating the self-attention of clients with heterogeneous
data may ruin the client-specific representations, potentially
degrading the model’s performance.

Unlike applying simple FedAvg operations on the whole
model, the above vanilla personalization mechanism can depict
client-specific self-attention layers by local training. However,
since these personalized self-attention layers are learned inde-
pendently without considering the potential inherent relations-
ships between clients, the obtained personalized self-attention
may be suboptimal. Moreover, the personalized self-attention
layers are not scalable as they increase linearly with
the increase in client numbers. Furthermore, the generalization
of personalized self-attention to novel clients is limited since
the whole self-attention layers must be retrained. Based on
this, we designed a novel transformer-based federated learning
framework called federated transformer personalization
(FedTP) that uses a learn-to-personalize mechanism instead of
the above vanilla personalization. Specifically, a hypernetwork
is learned on the server that generates projection matrices
in self-attention layers to produce clientwise queries, keys,
and values, while the other model parameters are aggregated
and shared. The learn-to-personalize mechanism for self-
attention layers through hypernetwork allows us to effectively
share parameters across clients and generate personalized self-
attention layers by learning a unique embedding vector for
each client. In addition to high accuracy, FedTP is also scalable
to the increase in client number and enjoys good generalization
capability to novel clients. Our main contributions are
summarized as follows.

1) We explore the effects of the self-attention mechanism
in personalized federated learning and we are the first
to reveal that FedAvg may have negative impacts on
self-attention where data heterogeneity is present. Based
on this, we propose a novel transformer-based federated
learning framework, namely, FedTP, which learns per-
sonalized self-attention for each client.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Another strategy is to add a proximal regularization term to handle client drift problems resulting from statistical heterogeneity. Examples include FedProx [23], pFedMe [24], and Ditto [25]. Instead, FedAlign [26] studied the data heterogeneity challenge of FL from the perspective of local learning generality rather than proximal restriction.

Instead of training a single global model, learning personalized models is more suitable for heterogeneous clients. One such approach is to seek an explicit tradeoff between the global model and the local models. Some researchers have considered interpolating the two models, e.g., L2GD [27] and LG-FEDAVG [28]. Similarly, Knn-Per [29] is also an interpolation of the two models, but the local model is built through a $k$-nearest neighbors method, which requires storing all the features of the samples. To increase the flexibility of this personalized model architecture for clients, some methods distill knowledge from a global teacher model into student models on the client devices. In this way, the clients learn a stronger personalized model. FedMD [30] and FedGen [31] are two examples of this approach. When there are inherent partitions or differences in the data distribution between clients, it can be more appropriate to train a federated learning model for each homogeneous group of clients through a clustering approach, such as clustered federated learning (CFL) [3], privacy-preserving federated adaptation (PFA) [32], or FedAMP [33], noting, though, that this may lead to high computation and communication overheads. FedPer [34] and FedRep [35] learn personalized classifier heads locally while sharing the base layers. FedBN [36] updates the client batch normalization layers locally, and pFedGP [37] learns a shared kernel function for all the clients and a personalized Gaussian process classifier for each client. In addition, some studies focus on specific fields and propose the corresponding PFL methods. Federated group synchronization (FEDGS) [38] is proposed to solve the heterogeneity caused by the rapidly changing streaming data and clustering nature of factory devices in the Industrial Internet of Things. FedStack [39] supports ensembling heterogeneous architectural client models in the area of healthcare. Furthermore, a horizontal federated reinforcement learning (HFRL)-based method [40] develops optimal electric vehicles charging/discharging control strategies for different EVs users to maximize their benefits. Unlike these works, our FedTP uses a learn-to-personalize mechanism to learn personalized self-attention layer in a transformer to better handle data heterogeneity among clients.

B. Transformers

The transformer model [8] was originally designed to improve the efficiency of machine translation tasks. It is a deep learning model based on a self-attention mechanism that has achieved state-of-the-art performance in many NLP tasks. Since it first emerged, many researchers have explored its applicability to vision tasks, with one of the most successful attempts being ViT [12]. As the first attempt to use transformer in federated learning, Park et al. [41] proposed a federated ViT for COVID-19 chest X-ray diagnosis. This framework leverages robust representations from multiple related tasks to improve the performance of individual tasks. Very recently, Qu et al. [11] conducted rigorous empirical experiments and showed that in federated learning settings, transformers are more suitable for situations with heterogeneous data distributions than CNNs. However, although transformer models have shown very promising performance in federated learning, most are only trained with the basic FedAvg algorithm, which may limit their true capabilities, especially in the non-IID scenarios. To see this architecture perform to its fullest potential, our FedTP learns a personalized transformer for each client. The self-attention mechanism captures data heterogeneity through a hypernetwork located on the server, which generates projection parameters in the self-attention layers.

C. Hypernetworks

Hypernetworks [42] are neural networks that can generate weights for a large target network with a learnable embedding vector as its input. pFedHN [43] was the first method to apply a hypernetwork to personalized federated learning, where a hypernetwork is learned on the server that generates personalized weights for the local CNNs on each client. Instead of generating all the model’s parameters, pFedLA [44] uses a hypernetwork on the server that outputs aggregated weights for each layer of the local model on each different client. Different from these two studies, Fed-RoD [45] learns a hypernetwork locally, which outputs personalized predictors for clients with the extra inputs of the clients’ class distributions. Notably, all these methods with hypernetworks are based on CNN architectures. In contrast, the hypernetwork in FedTP generates projection matrices in the self-attention layers of a transformer to produce clientwise queries, keys, and values.

III. FEDERATED LEARNING BY TRANSFORMER PERSONALIZATION

In this section, we present the design of our FedTP framework. Aiming to mitigate heterogeneity and build a high-quality personalized model for each client, FedTP learns personalized self-attention layers for each client.

A. Problem Formulation

The FedTP framework uses a traditional ViT [12] for image tasks and transformer [8] for language tasks. The input sequence $X$ is with a predetermined length of $m$, noting that images are partitioned into a sequence in the image preprocessing layer of ViT. This sequence is then transformed into a corresponding embedding matrix $H$ with a dimension of $\mathbb{R}^{m \times d}$. The queries, keys, and values of the self-attention mechanism are, respectively, denoted as $Q = HW^0$, $K = HW^k$, and $V = HW^v$. For convenience, we concatenate these projection parameters as $W = [W^0, W^k, W^v]$. Next, self-attention is applied through $\text{Attention}(Q, K, V) = \text{softmax}((QK^T / (d)^{1/2}))V$, where $d$ is the number of columns of $Q, K$, and $V$.

To simulate a federated scenario, we assume there are $N$ clients indexed by $[N]$, and each client $i$ has a local dataset $D_i = \{(x_{ij}^{(i)}, y_{ij}^{(i)})\}_{j=1}^{m_i} (1 \leq i \leq N)$ with $m_i$ samples drawn from a distinct data distribution $P_i$. Let $D = \bigcup_{i \in [N]} D_i$ denote the total datasets with the size of $M = \sum_{i=1}^{N} m_i$, and let
This is performed by a hypernetwork located on the server, which consists of simple MLP layers with the last layer being unique for each transformer block. Let \( \theta \) denote a personalized model for client \( i \), parameterized by \( \theta_i \). The optimization objective is

\[
\arg\min_{\Theta} \sum_{i=1}^{N} m_i \mathcal{L}_i(\theta_i)
\]

with \( \mathcal{L}_i(\theta_i) = \mathbb{E}_{(x_i^{(j)}, y_i^{(j)}) \in D_i} l(f(\theta_i; x_i^{(j)}), y_i^{(j)}) \), where \( \Theta = \{\theta_i\}_{i=1}^{N} \) is the set of personalized parameters, and \( l(\cdot, \cdot) \) denotes the per sample loss function common to all the clients. The loss function can be mean square error or cross entropy loss, which is selected according to different tasks.

### B. Vanilla Personalization for Self-Attention

Transformer-based federated learning has received increasing attention. It is well-known that the transformer can capture global interactions between the inputs by the self-attention layers. As analyzed in Introduction, simply applying the FedAvg operation on the self-attention layers of clients may degrade the model’s performance in data heterogeneity scenarios. Inspired by this, we first develop a personalized self-attention mechanism that maintains the self-attention layers of each client locally so as to personalize the model, while averaging other parameters to learn the common information. The left side of Fig. 2 shows the structure of vanilla personalization for self-attention.

Similar to FedAvg, these parameters are updated via local training for several epochs and aggregated on the server. Let \( W_i \) denote the projection parameters of the self-attention layer, and let \( \xi \) denote the parameters of other layers. The personalized parameter \( \theta_i \) is then split into \( \theta_i = \{W_i, \xi\} \). During the communication round \( t \), the personalized model \( f(\theta_i; \cdot) = f(W_i, \xi; \cdot) \) is trained locally for \( k \) steps, and the model becomes \( f(W_i^{t,k}, \bar{x}_i^{t,k}; \cdot) \), where \( W_i^{t,k} \) is retained locally to store the personalized information of client \( i \) and \( \bar{x}_i^{t,k} \) is aggregated across the clients via

\[
\bar{x}_i^t = \frac{1}{M} \sum_{i=1}^{N} m_i x_i^{t,k}.
\]

Thus, the objective function of FedTP can be derived from (1) to minimize the following loss:

\[
\arg\min_{\Theta} \sum_{i=1}^{N} \frac{m_i}{M} \mathcal{L}_i(W_i, \xi)
\]

where

\[
\mathcal{L}_i(W_i, \xi) = \frac{1}{M} \sum_{i=1}^{N} m_i \mathbb{E}_{(x_i^{(j)}, y_i^{(j)}) \in D_i} l\left(f\left(W_i, \xi; x_i^{(j)}\right), y_i^{(j)}\right).
\]

The above vanilla personalization procedure can generate personalized self-attention for each client by local training. However, since it ignores the potential inherent relationships between clients, the obtained personalized self-attention may be stuck in suboptimality. In addition, the personalized self-attention layers are not scalable as they increase linearly with the growing client numbers. Moreover, the generalization of personalized self-attention is not good. For example, when novel clients are involved, the model needs retraining to generate the specific self-attention layers for these novel clients.

### C. Learn-to-Personalize for Self-Attention

In this section, we develop FedTP that uses the learn-to-personalize mechanism to improve the vanilla personalization mechanism for self-attention. FedTP learns a hypernetwork [42] at the server to generate projection matrices in the self-attention layers for each client (see the right side of Fig. 2). By this way, FedTP can effectively share parameters among clients.

**Fig. 2.** Overview of FedTP. On the left, the self-attention layers are retained locally while the other parameters are aggregated on the server and shared among clients. The right shows the workflow of learn-to-personalize that generates the projection matrices in the self-attention layers of \( L \) transformer blocks.
across clients and maintain the flexibility of personalized transformers.

The hypernetwork on the server is denoted as \( h(\varphi; z_i) \) parameterized by \( \varphi \), where \( z_i \in \mathbb{R}^D \) can be a learnable embedding vector corresponding to client \( i \) or fixed. We implement the hypernetwork with simple fully connected layers and the last layer is unique for each transformer block. Given the embedding vector \( z_i \), the hypernetwork outputs the partial weights \( W_i = h(\varphi; z_i) \) for client \( i \), which is then decomposed into the projection parameters for the queries, keys, and values of the self-attention mechanism \( W_i = [W_i^Q, W_i^K, W_i^V] \).

By this way, the hypernetwork learns a category of projection parameters \( W_i = h(\varphi; z_i) [1 \leq i \leq N] \) for personalized self-attention. Hence, the personalized model is denoted as \( f(W_i, \xi; \cdot) = f(h(z_i; \varphi), \xi; \cdot) \), and the training loss in (3) is replaced by the following equation:

\[
\mathcal{L}_i(W_i, \xi) = \mathbb{E}_{(x_i^{(j)}, y_i^{(j)}) \in D_i} \left( f(h(\varphi; z_i), \xi; x_i^{(j)}), y_i^{(j)} \right). 
\]

Algorithm 1 demonstrates the procedures of the FedTP algorithm. We next introduce the update rules for model parameters in FedTP. First, in each local epoch \( k \) we update local model parameter \( \theta_i \) using stochastic gradient descent (SGD) by the following equation:

\[
\theta_i^{k+1} \leftarrow \theta_i^{k-1} - \alpha \nabla_{\theta_i} \mathcal{L}_i(\theta_i^{k-1}; B_i) \quad (6)
\]

where \( B_i \) is a mini-batch sampled from \( D_i \). Let \( C^t \) represent the set of sampled clients at each round \( t \). According to the chain rule, we can obtain the gradients of \( \varphi \) and \( z_i \) from (6):

\[
\nabla_{\varphi} \mathcal{L}_i = \sum_{i \in C^t} m_i \nabla_{\varphi} W_i^T \Delta W_i \quad (7)
\]

\[
\nabla_{z_i} \mathcal{L}_i = \sum_{i \in C^t} m_i \nabla_{z_i} W_i^T \Delta W_i \quad (8)
\]

where \( \Delta W_i = W_i^K - W_i^0 \) is the change in projection parameters after \( K \) epochs, and \( K \) is the local updating epochs. In communication round \( t \), we update the hypernetwork parameter \( \varphi \) and client embedding \( z_i \) using the calculated gradients

\[
\varphi = \varphi^{k-1} - \beta \nabla_{\varphi} \mathcal{L}_i^{k-1} \quad (9)
\]

\[
z_i = z_i^{k-1} - \beta \nabla_{z_i} \mathcal{L}_i^{k-1}. \quad (10)
\]

Compared with the vanilla personalization mechanism, *learn-to-personalize* for self-attention has several merits: 1) it can effectively share parameters across clients and better exploits the role of the self-attention mechanism in federated learning; 2) it offers scalability with the growing number of clients since the self-attention layer is generated by the shared hypernetwork with clientwise embedding vectors; and 3) it can be better generalized to novel clients whose data are unseen during training. The first and third points will be validated in Section IV. The second point can be easily shown by comparing the total number of parameters of *learn-to-personalize* and vanilla personalization. Based on the MLP structure of the hypernetwork, its total number of parameters is approximately equal to \( D_h \times D_v \), where \( D_h \) and \( D_v \) are the dimensions of the hidden layers in hypernetwork and the projection parameters of self-attention, respectively. In contrast, the total projection parameters of vanilla personalization increase linearly with the number of clients, i.e., \( N \times D_v \). When the client number \( N \) is larger than \( D_h \), *learn-to-personalize* for self-attention consumes less resource consumption.

D. Generalization Bound

We analyze the generalization bound of FedTP in this section. Before starting the analysis, we first introduce some assumptions as follows.

Assumption 1: We assume the embedding vectors \( z_i \) and the weights \( \varphi \) of hypernetwork \( h(\varphi, z_i) \) are bounded in a ball of radius \( R_h \), and the parameters of other layers \( \xi \) in transformer are bounded in a ball of radius \( R_t \). These assumptions can be denoted as follows:

\[
\|\varphi - \varphi'\| \leq R_h, \quad \|z_i - z_i'\| \leq R_h, \quad \|\xi - \xi'\| \leq R_t, \quad (11)
\]

Assumption 2 (Lipschitz Conditions): Let \( D_1, D_2, \ldots, D_N \) represent the real data distributions and \( \mathbb{E}_{D_j}(h(\varphi; z_i), \xi) = \mathbb{E}_{(x, y) \in D_j} f(h(\varphi; z_i), \xi; x, y) \) be the expected loss. We assume the following Lipschitz conditions hold:

\[
|\mathcal{L}_{D_j}(h(\varphi; z_i), \xi) - \mathcal{L}_{D_j}(h(\varphi; z_i), \xi')| \leq L_E \|\xi - \xi'\| \quad (12a)
\]

\[
|\mathcal{L}_{D_j}(h(\varphi; z_i), \xi) - \mathcal{L}_{D_j}(h(\varphi; z_i), \xi)| \leq L_h \|h(\varphi; z_i)\| \quad - h'(\varphi; z_i) \| \quad (12b)
\]

\[
\|h(\varphi; z_i) - h(\varphi; z_i')\| \leq L_p \|\varphi - \varphi'\| \quad (12c)
\]

\[
\|h(\varphi; z_i) - h(\varphi; z_i')\| \leq L_z \|z_i - z_i'\|. \quad (12d)
\]
Theorem 1: Suppose $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_N$ denote the empirical data distribution of $N$ clients with $\hat{\phi}$, $\hat{\xi}$, $\hat{\xi}$ the parameters learned by the corresponding empirical distributions. Denote $\mathcal{H}$ as the personalized hypothesis and let $d$ be the VC-dimension of $\mathcal{H}$. Suppose Assumptions 1 and 2 hold, with probability at least $1 - \delta$, we have
\begin{align}
\sum_{i=1}^{N} \frac{m_i}{M} \mathcal{L}_{\hat{D}_i}(h(\hat{\phi}; \hat{\xi}_i), \hat{\xi}) - \sum_{i=1}^{N} \frac{m_i}{M} \mathcal{L}_{D_i}(h(\phi^*; \xi^*_i), \xi^*) \\
\leq \sqrt{\frac{M}{2}} \frac{\log N}{\delta} + \sqrt{\frac{dN}{M}} \log \frac{eM}{d} + L_h \mathcal{R}_h(L_v + L_z) + L_z \mathcal{R}_z
\end{align}

where $\phi^*$, $\xi^*$, and $\xi^*$ represent the optimal parameters corresponding to the real distribution of each client, respectively; the size of the whole dataset is $M$ with the local data size of client $i$ being $m_i$.

Theorem 1 indicates that the performance of the model trained on the empirical distribution is affected by the complexity of the hypothesis class $\mathcal{H}$ (expressed by its VC-dimension), the number of clients, the size of the whole datasets, and the Lipschitz constants. The second part on the right-hand side of (13) can be formulated as $(\log(eM/d)/M/dN)^{1/2}$. It means that it is closely related to $(M/d)$. We denote the hypothesis class of FedTP with learn-to-personalize and with vanilla personalization by $\mathcal{H}_h$ and $\mathcal{H}_v$, respectively. The VC-dimension of $\mathcal{H}_h$ is smaller than the VC-dimension of $\mathcal{H}_v$ especially for large number of clients since we use one hypernetwork to generate the self-attention layers for all the clients with the learn-to-personalize mechanism. With the reduction of the VC-dimension $d$, $(\log(eM/d)/M/dN)^{1/2}$ will decrease. Thus, FedTP owns better generalization than vanilla personalization for self-attention especially when there are a large number of clients. The key lemmas and proof of Theorem 1 are given in the Appendix.

IV. EXPERIMENTS

In this section, we describe the setup of experiments, evaluate the performance of our model, and compare it to several baseline methods in various learning setups. In Section IV-A, we introduce benchmarks we compared, non-IID settings and model architectures we apply, and some implementation details. In the performance evaluation part, we mainly analyze the performance of different network backbones and different personalized parts of the transformer, generalization to novel clients, compatibility of FedTP to other methods, and visualization of attention maps. In Section IV-C, we study the effect of heterogeneity in label distribution and in noise-based feature imbalance, and the impact of different parameters. Fig. 3 shows the flowchart of our FedTP.

A. Experimental Setup

1) Benchmarks: We compared FedTP with the basic Federated algorithms, such as FedAvg [1] and FedProx [23], as well as with several advanced personalization algorithms: FedPer [34], pFedMe [24], FedBN [36], pFedHN [43], pFedGP [37], and FedRoD [45].

Fig. 3. Flowchart of FedTP. The blue part of the flowchart represents the operations that occur on the server side, while the green part indicates the client-side operations.

2) Non-IID Settings of Datasets: We use three popular benchmark datasets: CIFAR-10, CIFAR-100 [13], and Shakespeare [1], [46]. The first two are image datasets and the last one is a language dataset. For CIFAR-10 and CIFAR-100, we applied two split strategies to simulate the non-IID scenarios. The first is a “Pathological” setting, where each client is randomly assigned two/ten classes for CIFAR-10/ CIFAR-100 as in [24]. The sample rate on client $i$ of selected class $c$ is obtained by $a_{ic} / \sum_j a_{ijc}$, where $a_{ic} \sim U(4, 6)$. The second is a federated version created by randomly partitioning the datasets among clients using a symmetric Dirichlet distribution with parameter $\alpha = 0.3$, as in [29] and [47]. We create federated versions of CIFAR-10 by randomly partitioning samples with the same label among clients according to a Dirichlet distribution with parameter $\alpha = 0.3$. As for CIFAR-100, to create more realistic local datasets, we use a two-stage Pachinko allocation method to partition samples over “coarse” and “fine” labels. This method first generates a Dirichlet distribution with parameter $\alpha = 0.3$ over the coarse labels for each client, and then generates a Dirichlet distribution with parameter $\beta = 10$ over the coarse corresponding fine labels. For both the partitions, the classes and distribution of classes in the training and test sets of each client are the same. For Shakespeare, similar to the partitions in [29], we maintained the original split between the training and testing data specifically, 80% for training and 20% for testing. Table I summarizes the datasets, the corresponding tasks, and the number of clients and models.

3) Model Architectures: Similar to many federated works [43], [45], we adopted a ConvNet [48] with two convolutional layers and three fully connected layers as the local neural networks for baseline methods on CIFAR-10 and CIFAR-100. To improve the communication efficiency of
large-scale federated scenarios, we chose a tiny ViT with fewer parameters for FedTP, which consisted of eight blocks with eight self-attention layers in each block. The corresponding attention head number is 8, the patch size is 4, and the hidden size is 128. With regard to Shakespeare, we applied the same stacked two-layer LSTM model as [29] and [49] for benchmark methods. For a fair comparison with LSTM, we used a simple transformer [8] with two blocks as an encoder. The depth for both the LSTM and the simple transformer is 2, and the inner hidden dimension for both is 256. Since FedTP has a similar backbone for image classification tasks and language prediction tasks, we are able to use the same structure for the hypernetwork $h(\phi; \cdot)$ except for the last layer. We implement this hypernetwork similar to [43] with an MLP network and parameter mapping heads. The MLP network consists of four fully connected layers with 150 neurons, and each parameter mapping head is a single FC layer. For FedBN [36], we follow its original design which adds one BN layer after each convolution layer and fully connected layer (except for the last layer). For FedRoD [45], since the reported accuracy between the hypernetwork version and the linear version is similar, we adopt the linear version.  

4) Implementation Details: Following the experimental setting in pFedHN [43], we implement FedTP and the benchmark methods with 50 and 100 clients at 10% and 5% participation for CIFAR-10 and CIFAR-100, respectively. In the Shakespeare scenario, we treated each identity as a client and sampled 10% clients in each communication round. For the image classification task, we trained every algorithm for 1500 communication rounds. To make for an equivalent communication cost, pFedHN [43] is trained for 5000 global communication rounds. For the next-character prediction task, we trained the corresponding methods for 300 communication rounds. Both the tasks were optimized with an SGD optimizer for five local epochs with a default learning rate $\beta = 0.01$ and a batch size of $B = 64$. In FedTP and pFedHN, we optimize the hypernetworks using the SGD optimizer with a default learning rate $\beta = 0.01$. The server and all the clients are simulated on a cluster with an RTX 2080 Ti GPU, and all the algorithms are implemented in PyTorch [50].

B. Performance Evaluation

1) Evaluation Protocol: We tested each algorithm every five rounds during its last 200 global communication rounds and computed the mean test accuracy and standard deviation over these evaluation steps to estimate the model performance. The average test accuracy in each evaluation step is defined as the ratio of the sum of each client’s positive predictions over all the testing image numbers.

2) Performance Analysis: We compared the performance of FedTP with some well-known federated learning methods. Note that these methods were originally implemented based on CNN backbones. Here, we used the same transformer architecture of FedTP to replace their CNN backbones for a fair comparison. To mark the modification, we added a “-T” after the algorithm name to denote these methods. The average test accuracy of these algorithms is reported in Table II, which shows that FedTP outperforms all of them by a clear margin. This also validates our prior claims in the Introduction: 1) the FedAvg algorithm may impair the client-specific representations in transformers as Local-T works even better than FedAvg-T and 2) the personalized self-attention learned by FedTP can handle the data heterogeneity effectively. As shown in Table II, FedTP significantly outperforms Vanilla Personalized-T in all the settings, which verified that learn-to-personalize leverages the strengths of self-attention in transformer models to improve their performance in federated learning over heterogeneous data. In addition, FedTP is also superior to FedPer-T. This indicates that personalizing the projection matrices in self-attention layers is much more effective than personalizing the last classification head.

| #setting  | CIFAR-10          | CIFAR-100         |
|-----------|-------------------|-------------------|
|           | Pathological      | Dirichlet         | Pathological      | Dirichlet         |
|           | 50                | 100               | 50                | 100               |
| Local-T   | 84.55±0.15        | 82.21±0.08        | 69.94±0.13        | 66.68±0.13        | 55.91±0.17        | 49.23±0.11        | 27.87±0.12        | 23.34±0.10        |
| FedAvg-T  | 50.42±4.22        | 46.28±4.23        | 61.85±1.15        | 59.23±1.93        | 34.02±0.88        | 30.20±0.95        | 38.64±0.22        | 34.89±0.45        |
| FedProx-T [23] | 47.28±8.09        | 44.22±3.81        | 61.00±1.74        | 50.13±1.71        | 29.77±1.67        | 28.92±0.83        | 37.63±0.35        | 32.98±0.43        |
| FedPer-T [34] | 89.86±0.89        | 89.01±0.12        | 79.41±0.16        | 77.70±0.14        | 67.23±0.32        | 61.72±0.16        | 37.19±0.18        | 29.58±0.14        |
| pFedMe-T [24] | 82.26±0.61        | 77.57±0.52        | 71.45±0.87        | 68.13±0.67        | 53.08±0.72        | 39.94±0.91        | 33.59±0.52        | 25.95±0.64        |
| Vanilla Personalized-T | 86.97±0.07        | 84.90±0.11        | 75.14±0.12        | 72.33±0.16        | 61.98±0.13        | 57.60±0.12        | 37.82±0.11        | 32.81±0.11        |
| FedTP     | 90.31±0.26        | 88.39±0.14        | 81.24±0.17        | 80.27±0.28        | 68.05±0.24        | 63.76±0.39        | 46.35±0.29        | 43.74±0.39        |
TABLE III

| 
| RESULTS OF FEDTP AND THE BENCHMARK METHODS ON THE IMAGE DATASETS WITH DIFFERENT NON-IID SETTINGS |
| #setting | CIFAR-10 | CIFAR-100 |
| #client | Pathological | Dirichlet | Pathological | Dirichlet |
| 50 | 100 | 100 | 50 | 100 | 50 | 100 |
| FedAvg [1] | 47.79±0.48* | 44.12±1.10* | 56.59±0.91 | 57.52±1.01 | 15.71±0.35* | 14.59±0.40* | 18.16±0.58 | 20.34±1.34 |
| FedProx [23] | 50.81±2.94* | 57.38±1.08* | 58.51±0.65 | 56.46±0.66 | 19.39±0.63* | 21.32±0.71* | 19.18±0.30 | 19.40±1.76 |
| FedPer [34] | 83.39±0.47* | 80.99±0.71* | 77.99±0.02 | 74.21±0.07 | 48.32±1.46* | 42.08±0.14* | 22.06±0.39 | 20.06±0.26 |
| pFedMe [24] | 86.09±0.32* | 85.23±0.58* | 76.29±0.44 | 74.83±0.28 | 49.09±1.10* | 45.57±1.02* | 31.60±0.46 | 25.43±0.52 |
| FedBN [36] | 87.45±0.95 | 86.71±0.56 | 74.63±0.60 | 75.41±0.37 | 50.01±0.59 | 48.37±0.50 | 28.81±0.50 | 28.70±0.46 |
| pFedHN [43] | 88.38±0.29* | 87.97±0.70* | 71.79±0.57 | 68.36±0.86 | 59.48±0.67* | 53.24±0.31* | 34.05±0.41 | 29.87±0.69 |
| pFedGP [37] | 89.20±0.30* | 88.80±0.20* | — | — | 63.30±0.10* | 61.30±0.20* | — | — |
| FedRoD [45] | 89.87±0.03 | 89.05±0.04 | 75.01±0.09 | 73.99±0.09 | 56.28±0.14 | 54.96±1.30 | 27.45±0.73 | 28.29±1.53 |
| FedTP | 90.31±0.26 | 88.39±0.14 | 81.24±0.17 | 80.27±0.28 | 68.05±0.24 | 63.76±0.39 | 46.35±0.29 | 43.74±0.39 |

We use * to represent the results reported in previous works [37], [43] under the same experimental settings.

Fig. 4. Test accuracy and convergence behavior of FedTP and other transformer-based methods on CIFAR-100 with the Dirichlet setting over 50 clients.

TABLE IV

| AVERAGE TEST ACCURACY ON THE LANGUAGE DATASET |
| Shakespeare of Different Methods |
| Method | FedAvg | FedProx | FedPer | pFedMe | pFedTP |
| Test accuracy | 21.34±1.04 | 20.48±1.09 | 27.58±0.65 | 21.14±1.12 | 84.40±0.10 |

We analyzed the test accuracy versus global communication rounds curve of FedTP in comparison to other transformer-based methods in Fig. 4. FedTP shows a smoother curve and achieves higher accuracy compared with the others. This further proves that the self-attention mechanism is a crucial part to overcome data heterogeneity effectively.

3) Analysis of Network Backbone: We also implemented experiments to compare the performance of FedTP with many state-of-the-art federated learning methods that are based on CNN/LSTM. The average test accuracy of these algorithms for the image datasets and language dataset is shown in Tables III and IV, respectively. Clearly, FedTP outperforms the baseline methods in terms of the average test accuracy for most cases. It is worth noting that the hypernetworks in FedTP only produce the parameters of the self-attention layers, which take up around 9.8% parameters of the whole network. Although FedTP uses a much smaller ratio of personalized parameters, it still significantly outperforms pFedHN with an 11.31% improvement on average for CIFAR-100. As shown in Tables III and IV, changing the network backbone from a CNN or an LSTM to transformers will bring a significant improvement in the models’ ability to overcome data heterogeneity. Through comparing Tables II and III, the decrease in the accuracy of pFedMe-T and FedProx-T in some settings indicates that the performance of transformer-based models will be affected when there is a regularization term in these federated learning methods. This is possible because the regularization term may impair the self-attention mechanism, while our FedTP maintains the personalization of the self-attention layers through the learn-to-personalize mechanism. More details about the effects of learn-to-personalize will be shown in Section IV-B5.

4) Analysis of Different Personalized Parts: Here, we investigate the impacts of personalizing different components in the transformer: (1) the self-attention layers (our method), (2) the MLP layers, (3) the normalization layers, and (4) the whole encoder. In this experiment, we use the same hypernetwork to generate the parameters of these different components and keep the same structures of ViT as described in Section IV-A4. The results are shown in Table V, from which we can see that personalizing self-attention layers achieves the best results compared with personalizing other components. Besides, Table V also shows that personalizing the normalization layers works better than personalizing the MLP layers and the whole encoder.

5) Visualization of Attention Maps: We used attention roll-out [14] to visualize various transformer-based methods in federated learning. To aggregate information across self-attention
in each transformer block, we used the maximization operation instead of the average operation proposed in the original paper, and we discarded the least 30% attention values to filter out low-frequency signals. Figs. 5 and 6 present visual comparisons of attention maps between FedTP and other transformer-based variants on CIFAR-10 and CIFAR-100. Both Vanilla Personalized-T and FedTP exhibit client-specific self-attention for good visualization maps. In addition, our FedTP focuses more precisely on the critical parts of testing objects than pFedMe-T and FedAvg-T in many cases. This reflects that FedTP does well in depicting personal attention and again validates our claim.

6) Extension of FedTP: In this section, we mainly explore the compatibility of FedTP, and our FedTP is compatible with methods with personalized classifier head including FedPer [34], FedRod [45], and methods making use of local memory such as KNN-Per [29]. We inserted those modules into FedTP and evaluated their performance on the CIFAR-10/CIFAR-100 datasets. In our implementation, FedTP+FedPer retains each client’s own classification head locally based on FedTP. FedTP+FedRod computes the sum of the output of the personalized classification head and the global classification head as the prediction logits. FedTP+KNN establishes and maintains a local repository in a similar way to Knn-Per which relies on Faiss library [51]. The results are shown in Table VI.

Several discoveries can be found based on Fig. 7. On one hand, combining local memory to calibrate the classification head can improve the model’s performance, especially when the data quantity of each class is small. On the other hand,
TABLE VI
TEST ACCURACY ON AVERAGE FOR FEDTP AND ITS VARIANTS OVER 100 CLIENTS

| #setting       | CIFAR-10 Pathological | Dirichlet | CIFAR-10 Pathological | Dirichlet |
|----------------|------------------------|-----------|------------------------|-----------|
| FedTP (ours)   | 88.30±0.35             | 79.22±0.29| 60.90±0.26             | 39.74±0.32|
| FedTP+FedPer   | 89.42±0.10             | 78.18±0.14| 61.78±0.14             | 32.86±0.19|
| FedTP+FedRod   | 87.66±0.31             | 78.87±0.37| 64.62±0.22             | 42.30±0.26|
| FedTP+KNN      | 88.84±0.18             | 80.18±0.12| 64.30±0.27             | 40.70±0.26|

Fig. 7. Visualization of accuracy for FedTP extension experiments on the CIFAR-10 and CIFAR-100 datasets with 100 clients and 5% sample rate.

simply binding a personalized classification head to FedTP may diminish the accuracy when data comply with a Dirichlet distribution. This means a personal classifier may prevent FedTP from learning personalized self-attention when the data distribution is too diverse. Keeping the global head as FedTP+FedRod does will mitigate such problems and improve the model performance. To sum up, FedTP can well incorporate former algorithms to help each client better adapt its model to its data heterogeneity.

7) Generalization to Novel Clients: Similar to the settings in [43], we tested our method’s generalization ability against pFedMe, pFedHN, FedRod, and Vanilla Personalized-T on CIFAR-100 with the Dirichlet setting. Here, we sampled 20% of the clients as novel clients whose data are unseen during training, and only the personalized parameters of these models were fine-tuned for novel clients. Specifically, FedRod tuned the personalized parameters in the last classification layer and pFedMe learned all the parameters to get each client’s personalized model. Vanilla Personalized-T tuned clients’ personalized projection matrices of the self-attention layer. With regard to pFedHN and FedTP, the personalized parameters can choose between the clientwise embedding vectors with dimension 32 and the whole hypernetwork. The results are shown in Fig. 8, and we added “Embedding” and “Hypernetwork” after pFedHN and FedTP to distinguish between the two patterns. We can easily see that FedTP (Hypernetwork) is more robust and well adapts to novel clients in just a few epochs.

8) Analysis of Hypernetworks: To analyze the effects of hypernetworks, we compared FedTP with Vanilla Personalized-T, which restores each client’s projection parameters $W_i$ locally without using hypernetworks. Table II shows that FedTP has a prominent lead over Vanilla Personalized-T, indicating that hypernetworks indeed play an important role in FedTP. We further note that even if hypernetworks only produce the parameters of the self-attention layer, it is still good at encoding client-specific information into client embeddings $z_i$. The hypernetworks can simultaneously map the client embeddings $z_i$ into a manifold parameterized by hypernetwork parameters $\phi$.

Then we visualize the learned client embeddings by projecting them onto a 2-D plane using the t-SNE algorithm [52]. For convenience, we exploited “coarse” and “fine” labels of CIFAR-100 and split data in a special method similar to pFedHN [43]. In detail, we assigned each coarse class to five clients and then split the corresponding fine classes uniformly among those chosen clients. With this extra partition method, we trained FedTP and then visualized client embeddings after training. Fig. 9 shows that the learned personal embeddings of those clients with the same coarse labels are clustered together and they are mapped far away from those with different coarse labels. This result supports our claim that the hypernetwork is a highly effective way of encoding personalized information into their client embeddings $z_i$.

C. Ablation Study
1) Effect of Heterogeneity in Label Distribution: Data heterogeneity is the key problem that personalized federated learning aims to solve. We have shown that FedTP outperforms
various benchmark methods under several settings. Here, we concentrated on the label distribution heterogeneity. In previous experiments, we tested model performance with this heterogeneity by sampling class in each client following Dirichlet distribution with $\alpha = 0.3$. Now we explore more complete cases with $\alpha \in \{0.0, 0.1, 0.3, 0.5, 0.7, 0.9\}$ for the CIFAR-10 and CIFAR-100 datasets. The level of data heterogeneity is higher with a smaller $\alpha$. We set FedAvg-T as baseline and then compared FedTP with various personalized modules including FedPer-T, pFedHN [43], and FedBN [36].

From the results in Table VII, it can be easily found that with a higher degree of data heterogeneity, performance for FedAvg-T decreases while performance for methods with personalized modules increases. Among these methods, FedTP outperforms them in every case and is much more robust. Hence, FedTP could well overcome label distribution heterogeneity in a wide range. As $\alpha$ increases, some personalized federated learning algorithms may fail to make full use of heterogeneity in each client and even falls behind FedAvg-T in accuracy. FedTP, however, is still able to work well. This indicates that FedTP could better mine and exploit heterogeneity among clients even if heterogeneity is not so prominent.

2) Effect of Heterogeneity in Noise-Based Feature Imbalance: Generally, noise is another significant factor that may cause data heterogeneity. In the real-world, even though each client’s data distribution is similar, they may still suffer from different levels of noise and this will lead to feature imbalance among clients. Therefore, we explored the effect of such imbalance with a new partition method. For each client, we add increasing level of random Gaussian noise with mean $\mu = 0$, and each client’s standard deviation is derived by $\sigma_i = (\sigma_M/N - 1) \times i$, where $i \in \{0, 1, \ldots, N - 1\}$. In detail, we let the client number $N$ be 10 with 50% participation rate and conduct a series of experiments with $\sigma_M \in \{0, 5, 10, 15, 20\}$. We compared FedTP with various personalized federated methods. Fig. 10(a) demonstrates that our FedTP leads to other methods in all different cases. This means FedTP could deal with client-specific noise well. Furthermore, Fig. 10(b) shows

| Block number | CIFAR-10 | CIFAR-100 | CIFAR-10 | CIFAR-100 |
|--------------|----------|-----------|----------|-----------|
| Pathological | Dirichlet | Pathological | Dirichlet | Pathological | Dirichlet |
| 1            | 36.40±5.48 | 49.38±2.60 | 23.05±1.03 | 27.02±0.44 | 85.15±0.17 | 72.56±0.24 | 59.02±0.50 | 34.97±0.23 |
| 2            | 46.58±4.11 | 55.82±2.31 | 27.58±1.04 | 32.92±0.51 | 88.42±0.17 | 77.04±0.23 | 64.40±0.22 | 41.20±0.27 |
| 4            | 50.21±4.18 | 60.19±1.75 | 31.65±0.85 | 36.23±0.37 | 89.58±0.12 | 78.86±0.26 | 66.35±0.21 | 45.90±0.26 |
| 6            | 49.75±4.12 | 61.34±1.48 | 31.74±0.73 | 36.15±0.32 | 89.95±0.15 | 80.00±0.34 | 66.51±0.35 | 45.69±0.17 |
| 8            | 50.42±4.22 | 61.85±1.52 | 34.02±0.88 | 38.64±0.22 | 90.31±0.26 | 81.24±0.17 | 68.05±0.24 | 46.35±0.29 |
| 10           | 46.73±4.43 | 62.17±1.57 | 33.46±0.82 | 37.52±0.21 | 89.98±0.26 | 82.02±0.21 | 69.14±0.44 | 45.42±0.26 |

Table VII: Results of FedTP and the Benchmark Methods on the Image Datasets Over 100 Clients With Different $\alpha$ of the Dirichlet Setting.

Table VIII: Test Accuracy of FedTP and FedAvg-T on Image Datasets Over 50 Clients With Different Numbers of Self-Attention Blocks in ViT.

Fig. 10. (a) Test accuracy of FedTP and other benchmark methods on CIFAR-10 with different levels of noise. (b) We use different colors to present the different severity of noise, and the points denote the test accuracy of different clients in those different settings while the dotted line presents the average test accuracy of all the clients.
the average test accuracy of all the clients and the test accuracy of each client in FedTP for different levels of noise.

3) Impact of the Self-Attention Block Number: Here, we examined the impact of self-attention block number with its value \( L \in \{1, 2, 4, 6, 8, 10\} \). Table VIII shows results for FedTP with different block numbers. As can be seen, accumulating attention blocks will help the model develop a better ability to catch its data heterogeneity and improve model behavior to a certain extent. According to this table’s results, we choose eight as our FedTP’s default attention block number for CIFAR-10 and CIFAR-100.

4) Impact of Client Sample Rate: To explore how the number of participated client impacts the model performance, we implemented experiments with sample rate \( s \in \{0.05, 0.1, 0.15, 0.2, 0.25\} \). Fig. 11 clearly illustrates that the performance of FedAvg-T is obviously affected by the sampling rate while our FedTP is relatively more stable. This phenomenon also reflects the robustness of FedTP.

V. CONCLUSION

We have investigated the impact of self-attention under the federated learning framework and have revealed that FedAvg actually would degrade the performance of self-attention in the non-IID scenarios. To address this issue, we designed a novel transformer-based federated learning framework called FedTP that learns personalized self-attention for each client while aggregating the other parameters among the clients. Instead of using the vanilla personalization mechanism that maintains the self-attention layers of each client locally, we proposed the learn-to-personalize mechanism to further encourage cooperation among clients and to increase the scalability and generalization of FedTP. Specifically, learn-to-personalize is realized by learning a hypernetwork on the server that outputs the personalized projection matrices of self-attention layers to generate clientwise queries, keys, and values. Moreover, we also provided the generalization bound for FedTP with the learn-to-personalize mechanism.

Comprehensive experiments have verified that FedTP with the learn-to-personalize mechanism delivers great performance in the non-IID scenarios and outperforms the state-of-the-art methods in personalized federated learning. Since FedTP learns a central hypernetwork over all the clients, the model adapts better to novel clients, which confirms its better generalization to novel clients with the learn-to-personalize mechanism. When the local datasets suffer from different levels of noise, the performance of FedTP is better than other benchmarks in all the cases, which indicates that FedTP is more robust. Notably, FedTP enjoys good compatibility with many advanced personalized federated learning methods, i.e., we can simply combine FedTP with these methods, including FedPer, FedRod, KNN-Per, and so on to further enhance the model performance. It can be seen that these combined models have achieved better performance according to the experimental results. The primary limitation of FedTP is its slow training speed compared with the CNN-based federated learning models, a common issue of the transformer-based methods. A more computation-efficient and communication-efficient FedTP is under consideration for our future work.

APPENDIX

DETAILS ON GENERALIZATION BOUND

A. Key Lemmas

**Lemma 1 (McDiarmid’s Inequality [53]):** Let \( X_1, \ldots, X_n \) be independent random variables, where \( X_i \) has range \( \mathcal{X}_i \). Let \( f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathbb{R} \) be any function with the \( (c_1, \ldots, c_n) \)-bounded difference property: for every \( i = 1, \ldots, n \) and \( x_1, \ldots, x_i', x_i'' \in \mathcal{X}_i \times \cdots \times \mathcal{X}_n \), we have

\[
\sup_{x_i \in \mathcal{X}_i} |f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)| \leq c_i.
\]

Then for any \( \epsilon > 0 \)

\[
\mathbb{P}[f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)] \geq \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right).
\]

**Lemma 2 (Rademacher Complexity [53]):** Given a space \( A \) and a fixed distribution \( D_A \), let \( (a_1, \ldots, a_m) \) be a set of examples drawn i.i.d. from \( D_A \). Let \( \mathcal{F} \) be a class of functions \( f : A \to \mathbb{R} \), and the Rademacher complexity of \( \mathcal{F} \) is defined as follows:

\[
\mathcal{R}_{D_A}(\mathcal{F}) = \frac{1}{m} \mathbb{E}_{\sigma} \left[ \sup_{a \in A} \sum_{i=1}^m \sigma_i a_i \right]
\]

where \( \sigma_1, \ldots, \sigma_m \) are independent random variables uniformly chosen from \( \{-1, 1\} \).

B. Proof of Theorem 1

**Proof:** For the left side of Theorem 1, we have

\[
\left| \sum_{i=1}^N \frac{m_i}{M} \mathcal{L}_{\mathcal{D}_i}(h(\hat{\phi}; \hat{z}_i), \xi) - \sum_{i=1}^N \frac{m_i}{M} \mathcal{L}_{\mathcal{D}_i}(h(\phi^*; z_i^*), \xi^*) \right|
\]

\[
= \left| \sum_{i=1}^N \frac{m_i}{M} \left( \mathcal{L}_{\mathcal{D}_i}(h(\hat{\phi}; \hat{z}_i), \xi) - \mathcal{L}_{\mathcal{D}_i}(h(\phi^*; z_i^*), \xi^*) \right) \right|
\]

\[
+ \mathcal{L}_{\mathcal{D}_i}(h(\phi^*; z_i^*), \xi^*) - \mathcal{L}_{\mathcal{D}_i}(h(\phi^*; z_i^*), \xi^*) \right|
\]
where $N$ is the size of the whole dataset.

Using Lemma 2 and the results in [20], we can get

\[
\begin{aligned}
\mathbb{E}
\left[
\sum_{i=1}^{N} \frac{m_i}{M} \left(\mathcal{L}_{\mathcal{D}_i}(h(\psi_i^*; z_i^*), \xi^*) - \mathcal{L}_{\mathcal{D}_i}(h(\psi; z_i), \xi)\right)
\right]
\leq \mathbb{E}
\left[
\sum_{i=1}^{N} \frac{m_i}{M} \left(\mathcal{L}_{\mathcal{D}_i}(h(\psi_i^*; z_i^*), \xi^*) - \mathcal{L}_{\mathcal{D}_i}(h(\psi_i; z_i^*), \xi^*)\right)
\right] \leq \frac{d}{M} \log \frac{N}{\delta}.
\end{aligned}
\]
[15] W. Zhang et al., “R2Fed: Resilient reinforcement federated learning for industrial applications,” IEEE Trans. Ind. Informat., early access, Nov. 15, 2022, doi: 10.1109/TII.2022.3222369.

[16] W. Zhang et al., “Semi-asynchronous personalized federated learning for short-term photovoltaic power forecasting,” Digit. Commun. Netw., vol. 2022, pp. 1–10, Mar. 2022.

[17] L. Zhang, Y. Shi, Y.-C. Chang, and C.-T. Lin, “Federated fuzzy neural network with evolutionary rule learning,” IEEE Trans. Fuzzy Syst., vol. 31, no. 5, pp. 1653–1664, May 2023.

[18] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, “Towards personalized fed-learning,” IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 28, 2022, doi: 10.1109/TNNLS.2022.3160699.

[19] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, “Federated evaluation of on-device personalization,” 2019, arXiv:1910.10252.

[20] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, “Three approaches for personalization with applications to federated learning,” 2020, arXiv:2002.10619.

[21] A. Fallah, A. Mokhtari, and A. Ozdaglar, “Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 3557–3568.

[22] M. Khodak, M.-F. F. Balcan, and A. S. Talwalkar, “Adaptive gradient-based meta-learning methods,” in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019, pp. 5915–5926.

[23] T. Li, A. K. Sabu, M. Zehnder, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” in Proc. Mach. Learn. Syst., vol. 2, 2020, pp. 429–450.

[24] C. T. Dinh, N. Tran, and J. Nguyen, “Personalized federated learning with Moreau envelopes,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 21394–21405.

[25] T. Li, S. Hu, A. Beirami, and V. Smith, “Ditto: Fair and robust federated learning through personalization,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 6357–6368.

[26] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, “Local learning matters: Rethinking data heterogeneity in federated learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 8397–8406.

[27] F. Hanzlcy and P. Richtárik, “Federated learning of a mixture of global and local models,” 2020, arXiv:2002.05516.

[28] P. Pu Liang et al., “Think locally, act globally: Federated learning with local and global representations,” 2020, arXiv:2001.01523.

[29] O. Marfoq, G. Neglia, R. Vidal, and L. Kameni, “Personalized federated learning through local memorization,” in Proc. Int. Conf. Mach. Learn., 2022, pp. 15070–15092.

[30] D. Li and J. Wang, “FedMD: Heterogeneous federated learning via model distillation,” 2019, arXiv:1910.03581.

[31] Z. Zhu, J. Hong, and J. Zhou, “Data-free knowledge distillation for heterogeneous federated learning,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 12878–12889.

[32] B. Liu, Y. Guo, and X. Chen, “PFA: Privacy-preserving federated adaptation for effective model personalization,” in Proc. Web Conf., Apr. 2021, pp. 923–934.

[33] Y. Huang et al., “Personalized cross-silo federated learning on non-IID data,” in Proc. AAAI, 2021, pp. 7865–7873.

[34] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, “Federated learning with personalization layers,” 2019, arXiv:1912.00818.

[35] L. Collins, H. Hassan, A. Mokhtari, and S. Shakkottai, “Exploiting shared representations for personalized federated learning,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 2089–2099.

[36] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, “FedBN: Federated learning on non-IID features via local batch normalization,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–7.

[37] I. Achituve, A. Shamsian, A. Navon, G. Chechik, and E. Fetaya, “Personalized federated learning with Gaussian processes,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 8392–8406.

[38] Z. Li et al., “Data heterogeneity-robust federated learning via group client selection in industrial IoT,” IEEE Internet Things J., vol. 9, no. 18, pp. 17844–17857, Sep. 2022.

[39] T. Shaik et al., “FedStack: Personalized activity monitoring using stacked federated learning,” Knowl.-Based Syst., vol. 257, Dec. 2022, Art. no. 109929.

[40] Z. Zhang, Y. Jiang, Y. Shi, Y. Shi, and W. Chen, “Federated reinforcement learning for real-time electric vehicle charging and discharging control,” in Proc. IEEE Globecom Workshops, Dec. 2022, pp. 1717–1722.

[41] S. Park, G. Kim, J. Kim, B. Kim, and J. Chul Ye, “Federated split vision transformer for COVID-19 CXR diagnosis using task-agnostic training,” 2021, arXiv:2111.01338.

[42] D. Ha, A. Dai, and Q. V. Le, “HyperNetworks,” 2016, arXiv:1609.09106.

[43] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, “Personalized federated learning using hypernetworks,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 9489–9502.

[44] X. Ma, J. Zhang, S. Guo, and W. Xu, “Layer-wise model aggregation for personalized federated learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 10092–10101.

[45] H.-Y. Chen and W.-L. Chao, “On bridging generic and personalized federated learning,” in Proc. Int. Conf. Learn. Represent., 2022, pp. 1–32.

[46] S. Caldas et al., “LEAF: A benchmark for federated settings,” 2018, arXiv:1812.01097.

[47] S. Reddi et al., “Adaptive federated optimization,” 2020, arXiv:2003.00295.

[48] Y. LeCun, B. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[49] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, “Federated multi-task learning under a mixture of distributions,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 15434–15447.

[50] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019, pp. 8024–8035.

[51] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” 2017, arXiv:1702.08734.

[52] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, no. 11, pp. 1–10, 2008.

[53] M. Mohri, A. Rostamizadeh, and A. Talwalkar, Foundations of Machine Learning, Cambridge, MA, USA: MIT Press, 2018.
Jiangnan Tang received the B.E. degree in software engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2022. He is currently pursuing the M.E. degree in computer science with ShanghaiTech University, Shanghai, China. His current research interests include optimization algorithms and implicit neural networks.

Weiping Ding (Senior Member, IEEE) received the Ph.D. degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013. In 2016, he was a Visiting Scholar at the National University of Singapore, Singapore. From 2017 to 2018, he was a Visiting Professor at the University of Technology Sydney, Broadway, NSW, Australia. He is a Full Professor with the School of Information Science and Technology, Nantong University, Nantong, China. His main research directions involve deep neural networks, multimodal machine learning, and medical images analysis. He has published more than 80 IEEE Transactions papers. His 15 authored or coauthored papers have been selected as ESI Highly Cited Papers. Dr. Ding serves as an Associate Editor and an Editorial Board Member for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE/CAA JOURNAL OF AUTOMATICA SINICA, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON INTELLIGENT VEHICLES, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE, Information Fusion, Information Sciences, Neurocomputing, and Applied Soft Computing. He is the Leading Guest Editor for Special Issues in several prestigious journals, including IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON FUZZY SYSTEMS, and Information Fusion.

Chin-Teng Lin (Fellow, IEEE) received the bachelor’s degree in science from the National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1986, and the master’s and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1992, respectively. He is currently a Distinguished Professor and the Co-Director at the Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW, Australia. He is also the Honorary Chair Professor at Electrical and Computer Engineering, NCTU. He has authored or coauthored more than 425 journal articles, including about 200 IEEE journal articles in the areas of neural networks, fuzzy systems, brain–computer interface, multimedia information processing, cognitive neuro engineering, and human–machine teaming, which have been cited more than 32,000 times. Dr. Lin was elevated to be an IEEE Fellow for his contributions to biologically inspired information systems in 2005 and was elevated International Fuzzy Systems Association (IFSA) Fellow in 2012. He served as the Editor-in-Chief for the IEEE TRANSACTIONS ON FUZZY SYSTEMS and on the Board of Governors at IEEE Circuits and Systems (CAS) Society, IEEE Systems, Man, Cybernetics (SMC) Society, and IEEE Computational Intelligence Society.

Ye Shi (Member, IEEE) received the B.S. degree in statistics from Northwestern Polytechnical University, Taichang, China, in 2013 and the Ph.D. degree in electrical engineering with the University of Technology Sydney, Broadway, NSW, Australia, in 2018. He has been a Post-Doctoral Fellow with the School of Computer Science, University of Technology Sydney, since 2019. His current research interests include artificial intelligence, machine learning, computer vision, and smart grid. Dr. Shi received the Best Paper Award at the sixth IEEE International Conference on Control Systems, Computing and Engineering in 2016. He has been invited to be reviewers for many high-quality international journals including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON FUZZY SYSTEM, IEEE TRANSACTIONS ON SMART GRID, IEEE TRANSACTIONS ON POWER SYSTEM, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, Information Sciences, Applied Soft Computing, and IEEE SYSTEMS JOURNAL.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.