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. Recent works have shown that the self-attention mechanism in Transformer models is robust to distribution shifts. As such, 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. We investigated this relationship and revealed that federated averaging algorithms actually have a negative impact on self-attention where there is data heterogeneity. These impacts limit the capabilities of the Transformer model in federated learning settings. In this paper, 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 the vanilla personalization mechanism that maintains personalized self-attention layers of each client locally, we proposed the *learn-to-personalize* mechanism to further encourage the cooperation among clients and to increase the scablability and generalization of FedTP. Specifically, the *learn-to-personalize* is realized by learning a hypernetwork on the server that outputs the personalized projection matrices of self-attention layers to generate client-wise queries, keys and values. Furthermore, we present the generalization bound for FedTP with the *learn-to-personalize* mechanism. Notably, FedTP offers a convenient environment for performing a range of image and language tasks using the same federated network architecture – all of which benefit from Transformer personalization. Extensive experiments verify that FedTP with the *learn-to-personalize* mechanism yields state-of-the-art performance in non-IID scenarios.**

***Index Terms*—Personalized federated learning, Transformer, hypernetworks, learn to personalize, self-attention.**

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 communications 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.

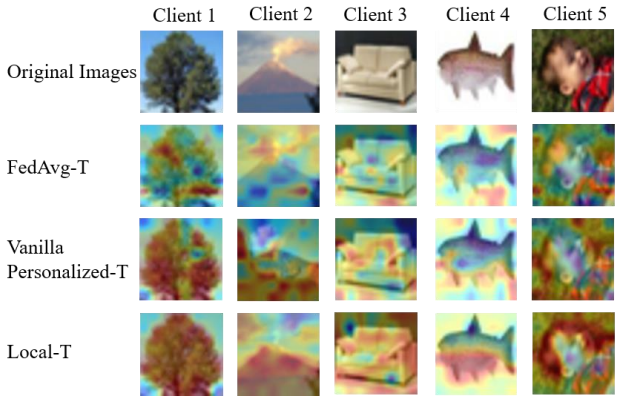


Fig. 1. An example showing the attention maps of Local-T, FedAvg-T and Vanilla Personalized-T over 5 clients with different local datasets. These results show that self-attention either trained locally or by a Vanilla personalized-T model can focus on task-specific information well. However, FedAvg-T disturbs these information.

Data heterogeneity is a common problem in the real world, since the training data collected from different clients varies heavily both in size and distribution [2]. Yet most federated learning frameworks are based on convolutional neural networks (CNNs), which generally focus on high-frequency local patterns that may be very sensitive to data heterogeneity [3]. Transformers [4], however, use self-attention to learn the global interactions between inputs [5] and, therefore, they tend to be more robust to distribution shifts and data heterogeneity [6]. Motivated by this, a very recent work [7] 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 has 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 problems of current personalized

Hongxia Li and Zhongyi Cai contributed equally to this work. Hongxia Li, Zhongyi Cai, Jingya Wang, Jiangnan Tang and Ye Shi are with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail:lihx2@shanghaitech.edu.cn, caizhy@shanghaitech.edu.cn, wangjingya@shanghaitech.edu.cn, tangjn2022@shanghaitech.edu.cn, shiye@shanghaitech.edu.cn).

Weiping Ding is with School of Information Science and Technology, Nantong University, Nantong 226019, China (e-mail: dwp9988@163.com)

Chin-Teng Lin is with the School of Computer Science, University of Technology Sydney, Broadway, NSW 2007, Australia (e-mail: chin-teng.lin@uts.edu.au).

federated learning are as follows:

1) The existing methods cannot deal with the problem of data heterogeneity and system heterogeneity between clients well, and most of them are based on convolutional neural networks, which are sensitive to non-independent and identically distributed data;
2) There is no unified federated learning framework for image and language tasks currently;
3) The current methods will have a great impact on the self-attention mechanism of Transformer in the aggregation process, and 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 [6]. 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 self-attention mechanisms. In our experiment, we compared the attention maps of three different Vision Transformer (ViT) models [8], these being: 1) Local-T, which trains a uniqe 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 [9] with five clients and using the Attention Rollout method [10] to produce attention maps. The original images and corresponding attention maps of these methods are shown in Fig. 1. We can see that both Local-T and 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 independently without considering the potential inherent relationships between clients, the obtained personalized self-attention may be sub-optimal. Moreover, the personalized self-attention layers are not scalable as they increase linearly with the increase of client numbers. Furthermore, the generalization of personalized self-attention to novel clients is limited since the whole self-attention layers must be re-trained. 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. Specificaaly, a hypernetwork is learned on the server that generates projection matrices in self-attention layers to produce client-wise 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 of client number and enjoys good generalization capability to novel clients. Our main contributions are summarized as follows:

1) We explore the effects of 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, that learns personalized 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 benchmark datasets under different non-IID data settings. The experimental results demonstrate that FedTP yields state-of-the-art performance over a wide range of personalized federated learning benchmark methods on both image and language tasks. It's worth noting that, FedTP offers a convenient environment for performing federated learning on both image and language tasks with the same network architecture benefiting from the Transformer personalization.

The rest of this paper 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 personalziation, including Vanilla personalization and *learn-to-personalize*; and the update of model parameters. Experimental results and discussions are provided in Section IV. Section V concludes the whole paper. 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. Currently, these methods can be divided into two main categories: global model personalization and learning personalized models [11]. Global model personalization 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 [12]–[15]. Another strategy is to add a proximal regularization term to handle client drift problems resulting from statistical heterogeneity. Examples include FedProx [16], pFedMe [17] and Ditto [18].

Instead of training a single global model, learning personalized models is more suitable for heterogeneous clients. One such approach is to seek an explicit trade-off between the global model and the local models. Some researchers have considered interpolating the two models, e.g., L2GD [19]

and LG-FEDAVG [20]. Similarly, Knn-Per [21] 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 feature of the samples. To increase the flexibility of this personalized model architecture for clients, some methods distil knowledge from a global teacher model into student models on the client devices. In this way, the clients learn a stronger personalized model. FedMD [22] and FedGen [23] 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, like CFL [24], PFA [25], or FedAMP [26], noting, though, that this may lead to high computation and communication overheads. FedPer [27] and FedRep [28] learn personalized classifier heads locally while sharing the base layers. FedBN [29] updates the client batch normalization layers locally, and pFedGP [30] learns a shared kernel function for all clients and a personalized Gaussian process classifier for each client. Unlike these works, our FedTP learns personalized self-attention to better handle data heterogeneity among clients.

B. Transformers

The Transformer model [4] 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 [8]. As the first attempt to use Transformer in federated learning, Park et al. proposed a federated ViT for COVID-19 chest X-ray diagnosis [31]. This framework leverages robust representations from multiple related tasks to improve the performance of individual tasks. Very recently, Qu et al. conducted rigorous empirical experiments and showed that, in federated learning settings, Transformers are more suitable for situations with heterogeneous data distributions than CNNs [7]. 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 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 [32] are neural networks that can generate weights for a large target network with a learnable embedding vector as its input. pFedHN [33] 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 [34] employs 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 [35] 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 client-wised 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 [8] for image tasks, and Transformer [4] 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 prepocessing 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 $Q = HW^Q$, $K = HW^K$ and $V = HW^V$. For convenience, we concatenated these projection parameters as $W = [W^Q, W^K, W^V]$. Next, self-attention is applied through $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V$, 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 $D_i = \{(x_i^{(j)}, y_i^{(j)})\}_{j=1}^{m_i} (1 \le i \le N)$ with m_i samples drawn from a distinct data distribution $\mathcal{P}_i$. 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 $f(\theta_i; \cdot)$ denote a personalized model for client i, parameterized by θ_i. The optimization objective is:

$$\arg\min_{\Theta} \sum_{i=1}^{N} \frac{m_i}{M} \mathcal{L}_i(\theta_i), \tag{1}$$

with $\mathcal{L}_i(\theta_i)$ formulated as

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{(x_i^{(j)}, y_i^{(j)})\in D_i} l(f(\theta_i; x_i^{(j)}), y_i^{(j)}), \tag{2}$$

where $\Theta = \{\theta_i\}_{i=1}^{N}$ is the set of personalized parameters, and $l(\cdot, \cdot)$ is the cross-entropy loss.

B. Vanilla personalization for self-attention

Transformer-based federated learning has received increasing attention. It is well known that Transformer can capture global interactions between the inputs by the self-attention layers. As we 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

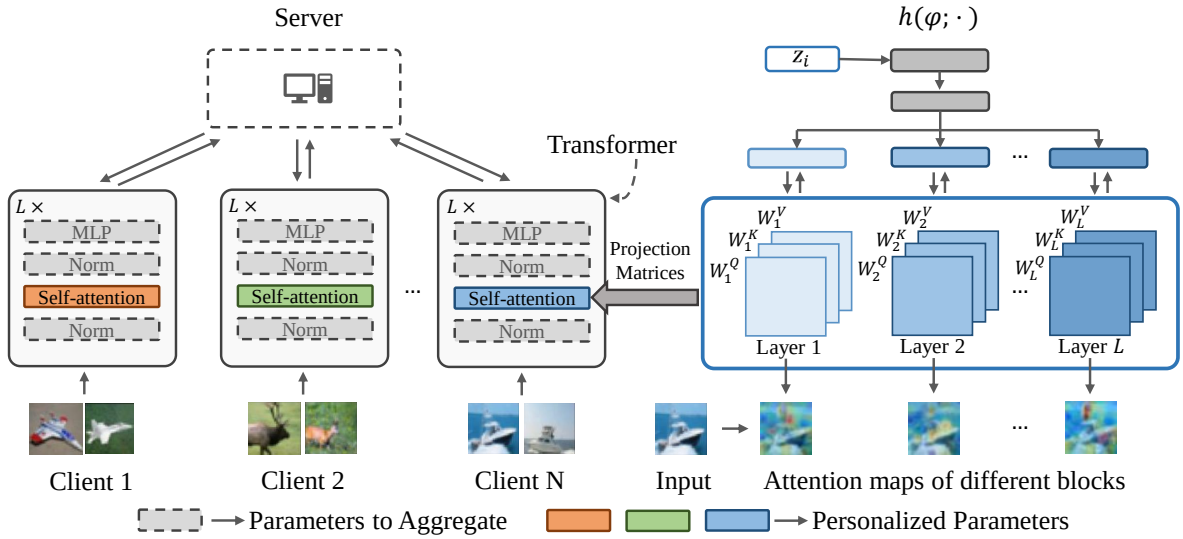


Fig. 2. An overview of FedTP. On the left, the self-attention layers are retained locally while the other parameters are aggregated on the server and shared among clients. The right shows the process of generating the projection matrices in the self-attention layers of L Transformer blocks. This is performed by a hypernetwork located on the server, which consists of simple MLP layers with the last layer being unique for each Transformer block.

of each client locally so as to personalize the model, while averaging other parameters to learn the common information.

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 ξ denote the parameters of other layers. The personalized parameter θ_i is then be 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 $f(W_i^{t,k}, \bar{\xi}_i^{t,k};\cdot)$, where $W_i^{t,k}$ is retained locally to store the personalized information of client i and $\bar{\xi}_i^{t,k}$ is aggregated across the clients via

$$\bar{\xi}^t = \sum_{i=1}^{N} \frac{m_i}{M} \xi_i^{t,k}. \tag{3}$$

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

$$\arg\min_{\Theta} \sum_{i=1}^{N} \frac{m_i}{M} \mathcal{L}_i(W_i, \xi), \tag{4}$$

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

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 sub-optimality. 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 re-training 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 utilizes the *learn-to-personalize* mechanism to improve the vanilla personalization mechanism for self-attention. FedTP learns a hypernetwork [32] at the server to generate projection matrices in the self-attention layers for each client (see Fig. 2). By this way, FedTP can effectively share parameters across clients and maintain the flexibility of personalized Transformers.

The hypernetwork on the server is denoted as $h(\varphi; z_i)$ parameterized by φ, 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 $\{W_i = h(\varphi; z_i) | 1 \le i \le N\}$ 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 Eq. (4) is replaced by

$$\begin{aligned}\mathcal{L}_i(W_i, \xi) &= \mathcal{L}_i(h(\varphi; z_i), \xi) \\ &= \sum_{i=1}^{N} \frac{m_i}{M} \mathbb{E}_{(x_i^{(j)}, y_i^{(j)}) \in D_i} l(f(h(\varphi; z_i), \xi; x_i^{(j)}), y_i^{(j)}).\end{aligned} \tag{5}$$

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

$$\theta_i^k \leftarrow \theta_i^{k-1} - \alpha \nabla_{\theta_i} \mathcal{L}_i(\theta_i^{k-1}; B_i), \tag{6}$$

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

Algorithm 1 Federated Transformer Personalization

Input: $T-$ number of communication rounds, $K-$ number of local epochs, $\alpha-$ learning rate of local update, $\beta-$ learning rate for global update.

Initialize parameters ξ, z and φ
for each communication rounds $t \in \{1, ..., T\}$ **do**
  Sample the set of clients $C^t \subset \{1, ..., N\}$
  for each client $i \in C^t$ **do**
    $\xi^{t,0} = \bar{\xi}^{t-1}$
    $W_i^{t,0} = h(\varphi^{t-1}; z_i^{t-1})$
    $\theta_i^{t,0} = \{W_i^{t,0}, \xi^{t,0}\}$
    for each local epoch $k \in \{1, ..., K\}$ **do**
      Sample mini-batch $B_i \in D_i$
      $\theta_i^{t,k+1} \leftarrow \theta_i^{t,k} - \alpha \nabla_{\theta_i} \mathcal{L}_i(\theta_i^{t,k}; B_i)$
    end for
    $\Delta W_i = W_i^{t,K} - W_i^{t,0}$
  end for
  $\bar{\xi}^t = \sum_{i \in C^t} \frac{m_i}{M} \xi^{t,K}$
  $\varphi^t = \varphi^{t-1} - \beta \sum_{i \in C^t} \frac{m_i}{M} \nabla_\varphi W_i^T \Delta W_i$
  $z_i^t = z_i^{t-1} - \beta \sum_{i \in C^t} \frac{m_i}{M} \nabla_{z_i} W_i^T \Delta W_i$
end for
return $\bar{\xi}^t$, φ^t and z_i^t

chain rule, we can obtain the gradients of φ and z_i from Eq. (6):

$$\nabla_\varphi \mathcal{L}_i = \sum_{i \in C^t} \frac{m_i}{M} \nabla_\varphi W_i^T \Delta W_i, \tag{7}$$

$$\nabla_{z_i} \mathcal{L}_i = \sum_{i \in C^t} \frac{m_i}{M} \nabla_{z_i} W_i^T \Delta W_i, \tag{8}$$

where $\Delta W_i = W_i^K - W_i^0$ is the change of projection parameters after K epochs, and K is the local updating epochs. In communication round t, we update hypernetwork parameter φ and client embedding z_i by using the calculated gradients:

$$\varphi^t = \varphi^{t-1} - \beta \nabla_\varphi \mathcal{L}_i^{t-1}, \tag{9}$$

$$z_i^t = z_i^{t-1} - \beta \nabla_{z_i} \mathcal{L}_i^{t-1}. \tag{10}$$

Compared to vanilla personalization mechanism, *learn-to-personalize* for self-attention has several merits: 1) it can effectively share parameters across clients and maintain the flexibility of personalized self-attention; 2) it offers greater scalability with the growing number of clients since the self-attention layer is generated by the shared hypernetwork with client-wise embedding vectors; 3) it can be better generalized to novel clients with far less computational costs as only the client-wise embedding vectors are fine-tuned.

D. Generalization Bound

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

Assmuption 1 *We assume the embedding vectors z_i and the weights φ of hypernetwork $h(\varphi, z_i)$ are bounded in a ball of radius R_h, and the parameters of other layers ξ in Transformer are bounded in a ball of radius R_t. These assumption can be denoted as:*

$$\|\varphi - \varphi'\| \leq R_h, \ \|z_i - z_i'\| \leq R_h, \ \|\xi - \xi'\| \leq R_t. \tag{11}$$

Assmuption 2 *(Lipschitz conditions) Let $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_N$ represent the real data distributions, and $\mathcal{L}_{\mathcal{D}_i}(h(\varphi; z_i), \xi) = \mathbb{E}_{(x,y)\in\mathcal{D}_i} l(f(h(\varphi; z_i), \xi; x), y)$ be the expected loss. We assume the following Lipschitz conditions hold:*

$$|\mathcal{L}_{\mathcal{D}_i}(h(\varphi; z_i), \xi) - \mathcal{L}_{\mathcal{D}_i}(h(\varphi; z_i), \xi')| \leq L_\xi \|\xi - \xi'\|, \tag{12a}$$

$$\begin{aligned} &|\mathcal{L}_{\mathcal{D}_i}(h(\varphi; z_i), \xi) - \mathcal{L}_{\mathcal{D}_i}(h'(\varphi; z_i), \xi)| \\ &\leq L_h \|h(\varphi; z_i) - h'(\varphi; z_i)\|, \end{aligned} \tag{12b}$$

$$\|h(\varphi; z_i) - h(\varphi'; z_i)\| \leq L_\varphi \|\varphi - \varphi'\|, \tag{12c}$$

$$\|h(\varphi; z_i) - h(\varphi; z_i')\| \leq L_z \|z_i - z_i'\|. \tag{12d}$$

Theorem 1 *Suppose $\hat{\mathcal{D}}_1, \hat{\mathcal{D}}_2, \cdots, \hat{\mathcal{D}}_N$ denote the empirical data distribution of N clients with $\hat{\varphi}$, $\hat{z}_i$, $\hat{\xi}$ the parameters learnt by the corresponding empirical distributions. Denote $\mathcal{H}$ as the personalized hypothesis and d be the VC-dimension of $\mathcal{H}$. Suppose Assumptions 1 and 2 hold, with probability at least $1 - \delta$, we have*

$$\begin{aligned} &\left| \sum_{i=1}^N \frac{m_i}{M} \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; \hat{z}_i), \hat{\xi}) - \sum_{i=1}^N \frac{m_i}{M} \mathcal{L}_{\mathcal{D}_i}(h(\varphi^*; z_i^*), \xi^*) \right| \\ &\leq \sqrt{\frac{M}{2} log \frac{N}{\delta}} + \sqrt{\frac{dN}{M} log \frac{eM}{d}} + L_h R_h (L_\varphi + L_z) \\ &+ L_\xi R_t, \end{aligned} \tag{13}$$

where φ^*, z_i^*, and ξ^* 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 $\sqrt{\frac{log(eM/d)}{M/dN}}$. It means that it is closely related to $\frac{M}{d}$. 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 clients with the *learn-to-personalize* mechanism. With the reduction of the VC-dimension d, $\sqrt{\frac{log(eM/d)}{M/dN}}$ 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.

TABLE I
DATASETS AND MODELS.

Dataset	Task	Clients	Total Samples	Model
CIFAR-10	Image Classification	50/100	60,000	ConvNet, ViT
CIFAR-100	Image Classification	50/100	60,000	ConvNet, ViT
Shakespeare	Next-character Prediction	683	2,578,349	LSTM, Transformer

TABLE II
THE RESULTS OF FEDTP AND THE BENCHMARK METHODS ON THE IMAGE DATASETS WITH DIFFERENT NON-IID SETTINGS.

	CIFAR-10				CIFAR-100			
#setting	Uniform		Dirichlet		Uniform		Dirichlet	
#client	50	100	50	100	50	100	50	100
FedAvg [1]	47.79±4.48*	44.12±3.10*	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 [16]	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 [27]	83.39±0.47*	80.99±0.71*	77.99±0.02	74.21±0.07	48.32±1.46*	42.08±0.18*	22.60±0.59	20.06±0.26
pFedMe [17]	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 [29]	87.45±0.95	86.71±0.56	74.63±0.60	75.41±0.37	50.01±0.59	48.37±0.56	28.81±0.50	28.70±0.46
pFedHN [33]	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 [30]	89.20±0.30*	88.80±0.20*	—	—	63.30±0.10*	61.30±0.20*	—	—
FedRoD [35]	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 (ours)	**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 [30], [33] under the same experimental settings.

A. Experimental Setup

1) Benchmarks: We compared FedTP with the basic Federated algorithms, such as **FedAvg** [1] and **FedProx** [16], as well as with several advanced personalization algorithms: **FedPer** [27], **pFedMe** [17], **FedBN** [29], **pFedHN** [33], **pFedGP** [30], and **FedRoD** [35].

2) Non-IID Settings of Datasets: We use three popular benchmark datasets: CIFAR-10, CIFAR-100 [9], and Shakespeare [1], [36]. 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 non-IID scenarios. The first is a "Uniform" setting, where each client is randomly assigned two/ten classes for CIFAR-10/CIFAR-100 as in [17]. The sample rate on client i of selected class c is obtained by $a_{i,c}/\sum_j a_{i,c}$, where $a_{i,c} \sim U(.4, .6)$. 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 [21], [37]. 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, in order to create more realistic local datasets, we use a two-stage Pachinko allocation method to partition samples over "coarse" and "fine" labels. This method firstly 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 partitions, the classes and distribution of classes in training and test set of each client are the same. For Shakespeare, similar to the partitions in [21], we maintained the original split between the training and testing data specifically, 80% for training and 20% for testing. Table I summarizes the datasets, corresponding tasks, and the number of clients and models.

3) Model Architectures: Similar to many federated works [33], [35], we adopted a ConvNet [38] with 2 convolutional layers and 3 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 8 blocks with 8 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 regards to Shakespeare, we applied the same stacked two-layer LSTM model as [21], [39] for benchmark methods. For a fair comparison with LSTM, we used a simple Transformer [4] 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 task and language prediction task, we are able to use the same structure for the hypernetwork $h(\varphi; \cdot)$ except for the last layer We implement this hypernetwork similar to [33] 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 [29], 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 [35], since the reported accuracy between hypernetwork version and linear version is similar, we adopt the linear version.

4) Implementation Details: Following the experimental setting in pFedHN [33], 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

TABLE III
AVERAGE TEST ACCURACY ON THE LANGUAGE DATASET SHAKESPEARE OF DIFFERENT METHODS.

Method	FedAvg	FedProx	FedPer	pFedMe	FedTP
Test accuracy	21.34±1.04	20.48±1.09	27.56±0.65	21.14±1.12	**84.40±0.10**

for 1500 communication rounds. To make for an equivalent communication cost, pFedHN [33] is trained for 5000 global communication rounds. For the next-character prediction task, we trained the corresponding methods for 300 communication rounds. Both tasks were optimized with a SGD optimizer for 5 local epochs with a default learning rate of $lr = 0.01$ and a batch size of $B = 64$. In FedTP and pFedHN, we optimize the hypernetworks by using the SGD optimizer with a default learning rate β = 0.01. The server and all clients are simulated on a cluster with an RTX 2080 Ti GPU, and all algorithms are implemented in PyTorch [40].

B. Performance Evaluation

1) Evaluation Protocol: We tested each algorithm on all datasets every 5 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 $\frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i}\sum_{j=1}^{m_i} Acc\left(f(\theta_i; x_i^j), y_i^j\right)$.

2) Performance Analysis: The average test accuracy of all algorithms for the image datasets and language dataset are reported in Table II and Table III, 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.

3) Generalization to Novel Clients: Similar to the settings in [33], we tested our method's generalization ability against pFedMe, pFedHN, and FedRod on CIFAR-10 with the Dirichlet setting. Here, we sampled 20% percent of the clients as novel clients whose data is unseen during training, and only the personalized parameters of these models were finetuned for novel clients. Specifically, FedRod tuned the personalized parameters in the last classification layer and pFedMe learned all parameters to get each client's personalized model. With regards to pFedHN and FedTP, the personalized parameters are the client-wise embedding vectors with dimension 32. From Fig. 3, we can easily see that FedTP is more robust and well adapts to novel clients in one epoch with the fewest tuning parameters.

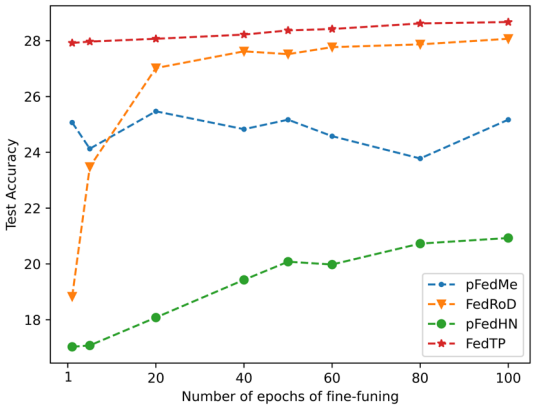


Fig. 3. Test accuracy of novel clients after fine-tuning the personalized parts of models trained on CIFAR-100.

4) Analysis of Network Backbone: As shown in Table II and Table III, 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. This is particularly efficient for the language dataset Shakespeare, where the averaged test accuracy increases dramatically by FedTP compared with other baselines. This also verifies our method's value in unifying vision classification tasks and sequence language prediction tasks in a more convenient environment for federated learning scenarios.

TABLE IV
AVERAGE TEST ACCURACY FOR FEDTP AND SEVERAL TRANSFORMER-BASED METHODS OVER 100 CLIENTS.

	CIFAR-10		CIFAR-100	
#setting	Uniform	Dirichlet	Uniform	Dirichlet
Local-T	82.21±0.08	66.68±0.13	49.25±0.11	23.34±0.10
FedAvg-T	46.28±4.23	59.23±1.93	30.20±0.95	34.89±0.45
FedProx-T	37.94±6.96	60.13±1.71	28.92±0.83	32.98±0.43
FedPer-T	**89.01±0.12**	77.70±0.14	61.72±0.16	29.58±0.14
pFedMe-T	77.57±0.52	68.13±0.67	39.94±0.91	25.95±0.64
FedTP (ours)	88.39±0.14	**80.27±0.28**	**63.76±0.39**	**43.74±0.39**

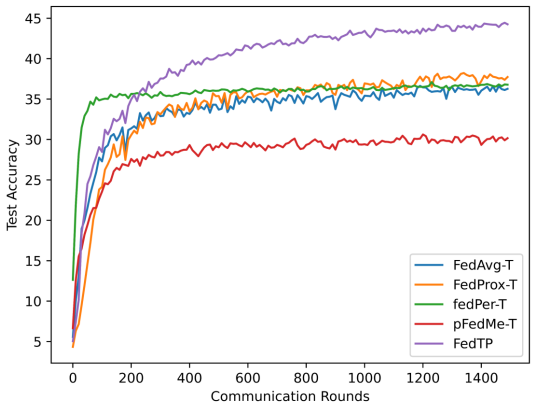


Fig. 4. The test accuracy and convergence behavior of FedTP and other Transformer-based Methods on CIFAR-100 with Dirichlet setting over 50 clients.

To eliminate the impact due to the differences in model architectures, we transferred the same Transformer-based architecture of FedTP to the baseline methods and further compared them with FedTP in Table IV. To denote these methods, we added a "-T" after the algorithm name. The

Fig. 5. Attention maps of FedTP and other Transformer-based variants implemented on CIFAR-10 over 50 clients.

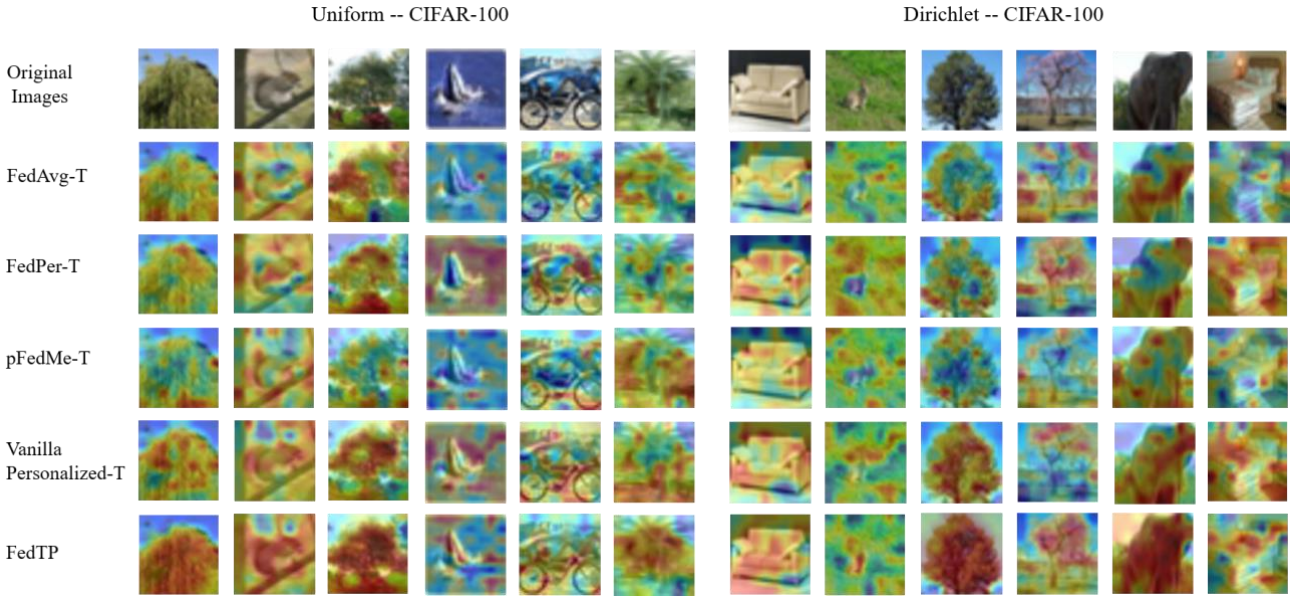


Fig. 6. Attention maps of FedTP and other Transformer-based variants implemented on CIFAR-100 over 50 clients.

results in Table IV show that, even if the same Transformer-based backbone is applied in the benchmark algorithms, our FedTP still outperforms them by a clear margin. This also validates our prior claims in the Introduction: 1) the FedAvg algorithm may ruin the client-specific representations in Transformers as Local-T works even better than FedAvg-T; 2) the personalized self-attention learned by FedTP can handle the data heterogeneity effectively. Additionally, 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.

We analyzed the test accuracy vs. global communication rounds curve of FedTP in comparison with 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.

5) Visualization of Attention Maps: We used Attention Rollout [10] to visualize various Transformer-based methods in federated learning. In order to aggregate information across self-attention in each Transformer block, we used the MAX operation instead of the AVG operation proposed in the original paper, and we discarded the least 30% attention values to filter out low-frequency signals. Fig. 5 and Fig. 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 Personalized-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 are compatible with methods with personalized classifier head including FedPer [27] and FedRod [35], and methods making use of local memory like KNN-Per [21]. 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 [41]. The results are shown in Table V.

TABLE V
TEST ACCURACY IN AVERAGE FOR FEDTP AND ITS VARIANTS OVER 100 CLIENTS.

	CIFAR-10		CIFAR-100	
#setting	Uniform	Dirichlet	Uniform	Dirichlet
Vanilla Personalized-T	84.90±0.11	72.33±0.16	57.60±0.12	32.81±0.11
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	78.67±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

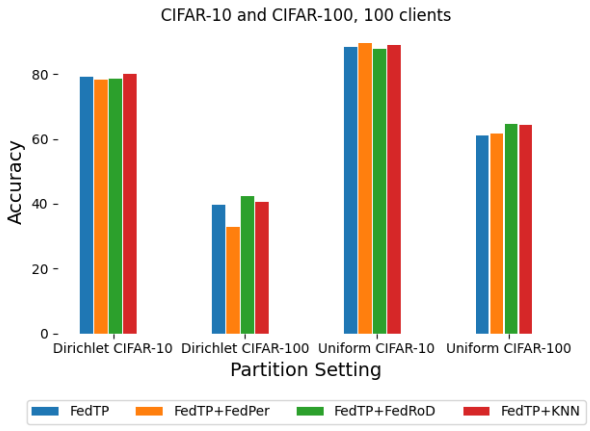


Fig. 7. Visualization of accuracy for FedTP Extension Experiments on CIFAR-10 and CIFAR-100 datasets with 100 clients and 5% sample rate.

Several discoveries can be found based on Fig. 7. On the 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, simply binding a personalized classification head to FedTP may diminish the accuracy when data complies 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 model performance. To sum up, FedTP can well incorporate former algorithms to help each client better adapt its model to its data heterogeneity.

7) 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 V shows that FedTP has a prominent lead over Vanilla Personalized-T, indicating that hypernetworks indeed play an important role in FedTP. We further notice 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 φ.

Then we visualize the learned client embeddings by projecting them onto a two-dimensional plane using the t-SNE algorithm [42]. For convenience, we exploited "coarse" and "fine" labels of CIFAR-100 and split data in a special method similar to pFedHN [33]. 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. 8 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.

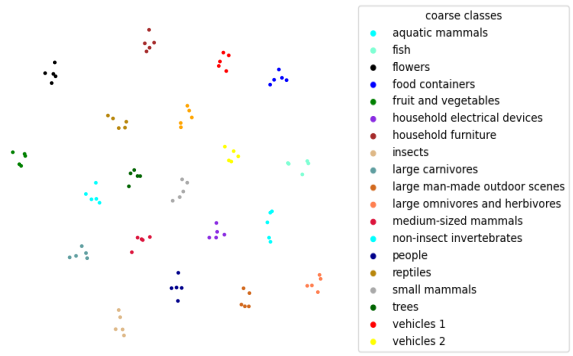


Fig. 8. t-SNE visualization of learned client embedding by FedTP on CIFAR-100 dataset.

C. Ablation Study

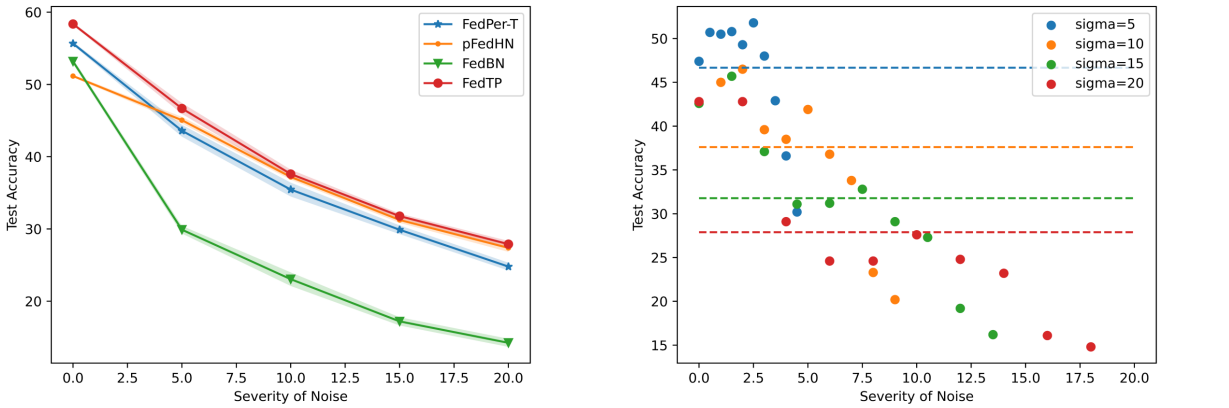
1) Effect of Heterogeneity in Label Distribution: Data heterogeneity is the key problem that Personalized federated learning aims to solve. We have shown that FedTP outperforms various benchmark methods under several settings. Here, we concentrated on the label distribution heterogeneity. In previous experiments, we tested model performance with this heterogeneity by sampling class in each client following Dirichlet distribution with α=0.3. Now we explore more complete cases

TABLE VI
THE RESULTS OF FEDTP AND THE BENCHMARK METHODS ON THE IMAGE DATASETS OVER 100 CLIENTS WITH DIFFERENT α OF DIRICHLET SETTING.

	CIFAR-10					CIFAR-100				
$\#\alpha$	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
FedAvg-T	40.99±6.20	59.23±1.93	63.69±1.33	65.29±1.12	65.82±0.82	32.72±0.81	34.89±0.45	36.25±1.79	36.99±0.44	37.51±0.33
FedPer-T	87.45±0.14	77.7±0.14	72.44±0.22	70.11±0.21	71.13±0.14	40.92±0.23	29.58±0.14	27.02±0.11	27.12±0.09	25.29±0.13
pFedHN	83.07±1.07	68.36±0.86	71.42±0.62	68.19±0.81	67.62±0.75	41.37±0.50	29.87±0.69	34.55±0.72	34.17±0.65	33.75±0.58
FedBN	84.93±0.53	75.41±0.37	71.79±0.63	69.57±0.56	68.70±0.47	33.+1±0.04	28.70±0.46	26.07±0.45	26.13±0.35	25.08±0.39
FedTP	**87.67±0.15**	**80.27±0.28**	**75.75±0.26**	**73.16±0.25**	**72.12±0.27**	**48.27±0.26**	**41.98±0.22**	**38.83±0.23**	**38.06±0.31**	**38.06±0.22**

TABLE VII
THE TEST ACCURACY OF FEDTP AND FEDAVG-T ON IMAGE DATASETS OVER 50 CLIENTS WITH DIFFERENT NUMBER OF SELF-ATTENTION BLOCKS IN VIT.

Block number	FedAvg-T				FedTP			
	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
	Uniform	Dirichlet	Uniform	Dirichlet	Uniform	Dirichlet	Uniform	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±0.30	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	43.90±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	43.69±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



(a) Test accuracy of different methods

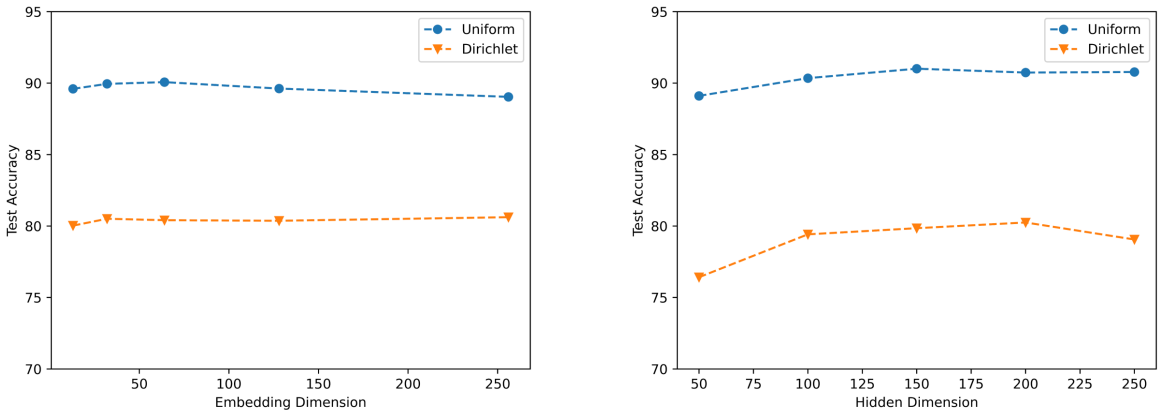
(b) Average test accuracy of all clients and test accuracy of each client in FedTP

Fig. 9. (a) The 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 presenting the average test accuracy of all clients.

with $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for CIFAR-10 and CIFAR-100 datasets. The level of data heterogeneity is higher with a smaller α. We set FedAvg-T as baseline and then compared FedTP with various personalized algorithms including FedPer-T, pFedHN [33], FedBN [29].

From the results in Table VI, 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 α 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 $\mu = 0$



(a) Dimension of Embedding Vectors

(b) Dimension of Hidden Layers

Fig. 10. The test accuracy on the two settings of CIFAR-10 over 50 clients showing the dimension of (a) the dimension of embedding vectors; (b) the dimension of hidden layers.

and each client's standard deviation is derived by $\sigma_i = \frac{\sigma_M}{N-1} * i$, where $i \in \{0, 1, \cdots, N-1\}$. 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. 9(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. 9(b) shows the average test accuracy of all 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 VII 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 8 as our FedTP's default attention block number for CIFAR-10 and CIFAR-100.

4) Impact of the Hypernetwork Size: We investigated the impact of the size of embedding vector and hidden dimension D on our FedTP by running the experiments with different embedding sizes $\{\lfloor 1 + \frac{n}{4} \rfloor, 32, 64, 128, 256\}$ and different $D \in \{50, 100, 150, 200, 250\}$. The results are shown in Fig. 10, from which we can find that the performance of FedTP is less affected by those two dimensions of Hypernetworks. This also indicates that our FedTP is robust. To achieve optimal performance, we fix the embedding dimension equal to 32 and $D = 150$.

5) Impact of Client Sample Rate: To explore how the number of participated client impacts 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.

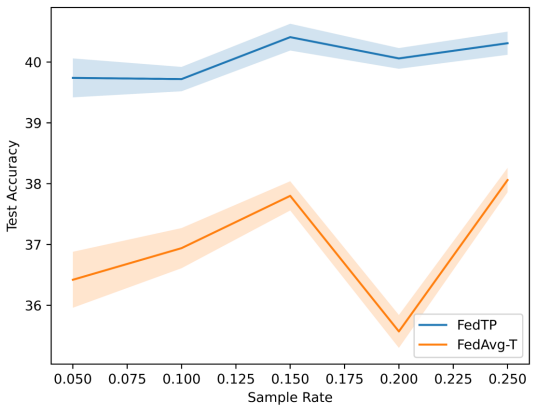


Fig. 11. The test accuracy of FedTP and FedAvg-T of Dirichlet setting on CIFAR-100 over 100 clients with different client sample rates.

V. Conclusions

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 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 the cooperation among clients and to increase the scablability and generalization of FedTP. Specifically, the *learn-to-personalize* is realized by learning a hypernetwork on the server that outputs the personalized projection matrices of self-attention layers to generate client-wise 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 non-IID scenarios and outperforms state-of-the-art methods in personalized federated learning. Since FedTP learns a central hypernetwork over all clients, the model adapts better to novel clients with the fewest tuning parameters, 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 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 and KNN-Per, etc to further enhance the model performance. It can be seen that these combined models have achieved better performance according to the experimental results. A more communication-efficient and more robust FedTP is under consideration for our future work.

APPENDIX: DETAILS ON GENERALIZATION BOUND

A. Key Lemmas

Lemma 1 *(McDiarmid's Inequality [43]) Let $X_1, \cdots, X_n$ 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 $(c_1, \cdots, c_n)-$bounded difference property: for every $i = 1, \cdots, n$ and $x_1, \cdots, x_n, x_i' \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, we have*

$$\sup_{x_i \in \mathcal{X}_i} |f(x_1, \cdots, x_i, \cdots, x_n) - f(x_1, \cdots, x_i', \cdots, x_n)| \leq c_i. \tag{14}$$

Then for any $\epsilon > 0$,

$$\begin{aligned} &\mathbb{P}[f(X_1, \cdots, X_n) - \mathbb{E}[f(X_1, \cdots, X_n)] \geq \epsilon] \\ &\leq \exp(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}). \end{aligned} \tag{15}$$

Lemma 2 *(Rademacher Complexity [43]) Given a space A and a fixed distribution D_A, let $\{a_1, \cdots, a_m\}$ 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:*

$$\Re_{D_A}(\mathcal{A}) = \frac{1}{m}\mathbb{E}_\sigma \left[\sup_{a \in A} \sum_{i=1}^m \sigma_i a_i\right]. \tag{16}$$

where $\sigma_1, \cdots, \sigma_m$ are independent random variables uniformly chosen from $\{-1, 1\}$.

B. Proof of Theorem 1

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

$$\begin{aligned} &\left|\sum_{i=1}^N \frac{m_i}{M}\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; \hat{z}_i), \hat{\xi}) - \sum_{i=1}^N \frac{m_i}{M}\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*; z_i^*), \xi^*)\right| \\ &= \left|\sum_{i=1}^N \frac{m_i}{M}\Big(\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; \hat{z}_i), \hat{\xi}) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*)\right. \\ &\left. + \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*) - \mathcal{L}_{\mathcal{D}_i}(h(\varphi^*; z_i^*), \xi^*)\Big)\right| \\ &\leq \left|\sum_{i=1}^N \frac{m_i}{M}(\mathcal{L}_{\hat{\mathcal{D}}_i}\Big(h(\hat{\varphi}; \hat{z}_i), \hat{\xi}) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*)\Big)\right| \\ &+ \left|\sum_{i=1}^N \frac{m_i}{M}\Big(\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*) - \mathcal{L}_{\mathcal{D}_i}(h(\varphi^*; z_i^*), \xi^*)\Big)\right|. \end{aligned} \tag{17}$$

The objective function is split into two parts, and then we will bound them separately. For the first part in Eq. (17), suppose Assumptions 1 and 2 hold, we get

$$\begin{aligned} &\left|\sum_{i=1}^N \frac{m_i}{M}\Big(\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; \hat{z}_i), \hat{\xi})\Big)\right| \\ &= \left|\sum_{i=1}^N \frac{m_i}{M}\Big(\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; z_i^*), \xi^*)\right. \\ &+ \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; z_i^*), \xi^*) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; \hat{z}_i), \xi^*) \\ &\left. + \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; \hat{z}_i), \xi^*) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi}; \hat{z}_i), \hat{\xi})\Big)\right| \\ &\leq \sum_{i=1}^N \frac{m_i}{M}\Big(L_h\|h(\varphi^*; z_i^*) - h(\hat{\varphi}; z_i^*)\| \\ &+ L_h\|h(\hat{\varphi}; z_i^*) - h(\hat{\varphi}; \hat{z}_i)\| + L_\xi\|\xi^* - \hat{\xi}\|\Big) \\ &\leq \sum_{i=1}^N \frac{m_i}{M}\Big(L_h L_\varphi\|\varphi^* - \hat{\varphi}\| + L_h L_z\|z_i^* - \hat{z}_i\| + L_\xi\|\xi^* - \hat{\xi}\|\Big) \\ &= L_h L_\varphi\|\varphi^* - \hat{\varphi}\| + L_\xi\|\xi^* - \hat{\xi}\| + \sum_{i=1}^N \frac{m_i}{M}L_h L_z\|z_i^* - \hat{z}_i\| \\ &\leq L_h L_\varphi R_h + L_\xi R_t + L_h L_z R_h \\ &= L_h R_h(L_\varphi + L_z) + L_\xi R_t. \end{aligned} \tag{18}$$

For the second part in Eq. (17), by replacing $f(\cdot)$ with $\sum_{i=1}^N \Big(\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*; z_i^*), \xi^*) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*)\Big)$ in Lemma 1, and let $\delta = \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right)$, with probability at least $1-\delta$, the following inequality holds,

$$\begin{aligned} &\sum_{i=1}^N \frac{m_i}{M}\Big(\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*; z_i^*), \xi^*) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*)\Big) \\ &\leq \mathbb{E}\left[\sum_{i=1}^N \frac{m_i}{M}\Big(\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*; z_i^*), \xi^*) - \mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*; z_i^*), \xi^*)\Big)\right] \\ &+ \sqrt{\frac{M}{2}log\frac{N}{\delta}}, \end{aligned} \tag{19}$$

where M is the size of the whole dataset.

Utilizing Lemma 2 and the results in [13], we can get

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{N}\frac{m_i}{M}\Big(\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*;z_i^*),\xi^*)-\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*;z_i^*),\xi^*)\Big)\right]\\
&\leq \mathbb{E}\left[\sum_{i=1}^{N}\frac{m_i}{M}\Big(\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*;z_i^*),\xi^*)-\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*;z_i^*),\xi^*)\Big)\right]\\
&\leq \sum_{i=1}^{N}\frac{m_i}{M}\Re_{\mathcal{D}_i}(\mathcal{H})\leq\sum_{i=1}^{N}\frac{m_i}{M}\sqrt{\frac{dN}{m_i}log\frac{em_i}{d}}\\
&\leq \sum_{i=1}^{N}\frac{m_i}{M}\sqrt{\frac{dN}{m_i}log\frac{eM}{d}}\leq\sqrt{\frac{dN}{M}log\frac{eM}{d}}.
\end{aligned}\tag{20}$$

Therefore, we can get

$$\begin{aligned}
&\left|\sum_{i=1}^{N}\frac{m_i}{M}\Big(\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*;z_i^*),\xi^*)-\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\varphi^*;z_i^*),\xi^*)\Big)\right|\\
&\leq\sqrt{\frac{M}{2}log\frac{N}{\delta}}+\sqrt{\frac{dN}{M}log\frac{eM}{d}}.
\end{aligned}\tag{21}$$

Summarizing the results above, we can get

$$\begin{aligned}
&\left|\sum_{i=1}^{N}\frac{m_i}{M}\mathcal{L}_{\hat{\mathcal{D}}_i}(h(\hat{\varphi};\hat{z}_i),\hat{\xi})-\sum_{i=1}^{N}\frac{m_i}{M}\mathcal{L}_{\mathcal{D}_i}(h(\varphi^*;z_i^*),\xi^*)\right|\\
&\leq\sqrt{\frac{M}{2}log\frac{N}{\delta}}+\sqrt{\frac{dN}{M}log\frac{eM}{d}}+L_hR_h(L_\varphi+L_z)\\
&+L_\xi R_t.
\end{aligned}\tag{22}$$

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[2] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[3] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[6] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10231–10241, 2021.

[7] L. Qu, Y. Zhou, P. P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, and D. Rubin, "Rethinking architecture design for tackling data heterogeneity in federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10061–10071, 2022.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[9] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[10] S. Abnar and W. Zuidema, "Quantifying attention flow in transformers," *arXiv preprint arXiv:2005.00928*, 2020.

[11] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

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

[13] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[14] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[15] M. Khodak, M.-F. F. Balcan, and A. S. Talwalkar, "Adaptive gradient-based meta-learning methods," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[17] C. T Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21394–21405, 2020.

[18] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *International Conference on Machine Learning*, pp. 6357–6368, PMLR, 2021.

[19] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," *arXiv preprint arXiv:2002.05516*, 2020.

[20] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Federated learning with local and global representations," *arXiv preprint arXiv:2001.01523*, 2020.

[21] O. Marfoq, G. Neglia, R. Vidal, and L. Kameni, "Personalized federated learning through local memorization," in *International Conference on Machine Learning*, pp. 15070–15092, PMLR, 2022.

[22] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[23] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *International Conference on Machine Learning*, pp. 12878–12889, PMLR, 2021.

[24] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[25] B. Liu, Y. Guo, and X. Chen, "Pfa: Privacy-preserving federated adaptation for effective model personalization," in *Proceedings of the Web Conference 2021*, pp. 923–934, 2021.

[26] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data.," in *AAAI*, pp. 7865–7873, 2021.

[27] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[28] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *International Conference on Machine Learning*, pp. 2089–2099, PMLR, 2021.

[29] X. Li, M. JIANG, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," in *International Conference on Learning Representations*, 2020.

[30] I. Achituve, A. Shamsian, A. Navon, G. Chechik, and E. Fetaya, "Personalized federated learning with gaussian processes," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8392–8406, 2021.

[31] S. Park, G. Kim, J. Kim, B. Kim, and J. C. Ye, "Federated split vision transformer for covid-19cxr diagnosis using task-agnostic training," *arXiv preprint arXiv:2111.01338*, 2021.

[32] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[33] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized federated learning using hypernetworks," in *International Conference on Machine Learning*, pp. 9489–9502, PMLR, 2021.

[34] X. Ma, J. Zhang, S. Guo, and W. Xu, "Layer-wised model aggregation for personalized federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10092–10101, 2022.

[35] H.-Y. Chen and W.-L. Chao, "On bridging generic and personalized federated learning," *International Conference on Learning Representations*, 2022.
[36] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.
[37] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.
[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[39] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15434–15447, 2021.
[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
[41] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *CoRR*, vol. abs/1702.08734, 2017.
[42] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.
[43] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.