# Federated Learning with Personalization Layers

MANOJ GHUHAN ARIVAZHAGAN, Adobe Research
VINAY AGGARWAL, Indian Institute of Technology, Roorkee, India
AADITYA KUMAR SINGH, Indian Institute of Technology, Kharagpur, India
SUNAV CHOUDHARY, Adobe Research

The emerging paradigm of federated learning strives to enable collaborative training of machine learning models on the network edge without centrally aggregating raw data and hence, improving data privacy. This sharply deviates from traditional machine learning and necessitates design of algorithms robust to various sources of heterogeneity. Specifically, statistical heterogeneity of data across user devices can severely degrade performance of standard federated averaging for traditional machine learning applications like personalization with deep learning. This paper proposes FedPer, a base + personalization layer approach for federated training of deep feed forward neural networks, which can combat the ill-effects of statistical heterogeneity. We demonstrate effectiveness of FedPer for non-identical data partitions of CIFAR datasets and on a personalized image aesthetics dataset from Flickr.

Additional Key Words and Phrases: On Device AI, Edge Computing, Personalization

**ACM Reference format:**
Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated Learning with Personalization Layers. 1, 1, Article 1 (December 2019), 18 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Modern day mobiles, wearables, home assistants and other internet enabled devices possess sophisticated hardware and software capabilities to support a variety of applications and generate


Manuscript submitted to ACM

large volumes of contextual and user data. Owing to privacy concerns and increasing compute and storage capabilities at the network edge, it has become increasingly attractive to keep data locally on user/client/edge devices and execute machine learning (ML) model training computations on device with locally available data and occassional communication with an aggregating parameter server. This approach to training ML models is termed as *federated learning* [**?** ] and is in sharp contrast to traditional ML training and prediction pipelines that rely on central aggregation of raw data.

The recent overview article by **?** ] articulates the many unique challenges faced by a federated learning system. One such challenge is that the effective data distribution at different clients can vary greatly across the (potentially millions) of participating devices. Such *statistical heterogeneity* can be detrimental to the performance of ML training algorithms for several applications like personalization, recommendation, fraud detection, *etc.* since traditional ML training algorithms are designed for central or distributed computation environments where data partitioning can be tightly controlled. Overcoming the ill-effects of statistical heterogeneity in federated learning is an active area of research with several recent works [**? ? ? ?** ].

Herein, we study the effects of the personalization setup as a source of statistical heterogeneity in federated learning with deep feedforward neural networks. Personalization is a key application of ML and is possible because users differ in their preferences as captured from raw user data. In a federated setup where edge devices belong to users, this necessarily means that data for personalization is statistically heterogeneous. For personalization via recommendation, **?** ] present the first attempt to extend collaborative filtering to a federated setup and **?** ] design a meta-learning approach for federated training. While these approaches were demonstrated to perform very well on the MovieLens 100k dataset [**?** ], it isn't obvious how to extend them to deep neural network models or to problems that cannot be addressed by collaborative filtering. We believe that the right approach to personalized federated learning is a highly non-trivial question on which the research community has barely scratched the surface. In particular, the challenge comes from multiple facets, including but not limited to:

- Many personalization tasks like personalized image aesthetics [**?** ] and personalized highlight detection [**?** ] have no explicit user features in the data and need to extract such features during training. Hence, same input data can receive different labels from different users implying that the personalized models must differ across users to be able to predict different labels for similar test data. This is already outside the scope of standard federated learning [**?** ] which learns one global model and effectively replicates it locally on every client.
- The number of training samples per user is not large enough to train individual ML models in isolation. Hence, some communication and collaboration is needed to leverage *wisdom of*

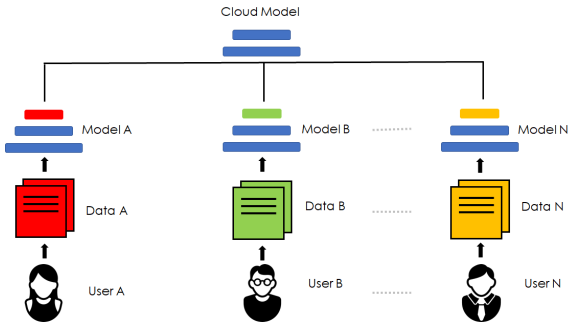



Fig. 1. Pictorial view of proposed federated personalization approach. All user devices share a set of base layers with same weights (colored blue) and have distinct personalization layers that can potentially adapt to individual data. The base layers are shared with the parameter server while the personalization layers are kept private by each device.

*the crowd*. However, collaborative filtering algorithms may not be applicable since they require considerable overlap between items rated by different users. According to **? ?** ], the sufficient overlap condition fails in typical datasets for personalized image aesthetics and personalized highlight detection among other tasks.

### 1.1 Contributions

(1) We propose to capture personalization aspects in federated learning by viewing deep learning models as *base* + *personalization* layers as illustrated in Figure 1. Our training algorithm comprises of the base layers being trained by federated averaging (or some variant thereof) and personalization layers being trained only from local data with stochastic gradient descent (or some variant thereof). We demonstrate that the personalization layers that are free from the federated averaging (FedAvg) procedure can help combat the ill-effects of statistical heterogeneity.

(2) We demonstrate that standard federated learning setup is highly unsuitable for personalization tasks by comparing its performance against our federated personalization approach *w.r.t.* two datasets: (a) non-identically partitioned CIFAR-10/CIFAR-100 dataset, and (b) FLICKR-AES dataset from **?** ]. Interestingly, standard federated learning's failure on the FLICKR-AES dataset is qualitatively and quantitatively very different from its poor performance on the non-identically partitioned CIFAR-100 dataset.

## 2 RELATED WORK

We have built upon several ideas in literature for the research communicated in this paper. We call out the overlaps and differences below.

**Federated Learning:** This paradigm for training ML models on user/client/edge devices was first proposed through the papers **? ? ?** ]. The initial focus was on reducing the communication overhead while maintaining the statistical performance of the learned global model. Since these early works, researchers have worked to shed light on the other challenges of making a federated system work (which include *system heterogeneity* and *privacy leakage* besides *statistical heterogeneity* and *communication overhead*) and potential solutions in various application scenarios. These are highlighted with bibliographic references in the recent overview article **?** ]. While we acknowledge that progress on all four challenges are needed for practically realizing federated learning systems, *our novelty in this work is restricted to the statistical heterogeneity aspect while maintaining composability with the current approaches to tackle the other three challenges*. Our 'base + personalization layers' notion for federated deep learning effectively learns different overlapping local models to better capture statistical heterogeneity. This approach expands on the original scope of federated learning which was to train one global model and essentially replicate it on all user devices. It is known that the original FedAvg algorithm is possibly non-convergent when training on pathological and/or highly non-identical data partitions [**?** **?** ].

**Multi-task and Transfer Learning:** Modeling and algorithmic techniques in multi-task and transfer learning domains try to learn with lesser training data by leveraging latent relationships across multiple tasks defined on the same dataset. **?** ] were the first to recognize that multi-task learning models could be effectively used to study various heterogeneities of a federated learning setup and incorporate aspects of personalization of local models. The authors restrict consideration to convex/bi-convex formulations of multi-task learning which cannot be easily extended to neural network models. Since deep learning models naturally yield to multi-task formulations due to their layered structure [**?** ], it should be possible to study federated deep multi-task learning setups that generalize assumptions in **?** ]. This is indeed the primary inspiration for the 'base + personalization layers' formulation proposed in this paper. Intuitively, the base layers can act as the shared layers in a multi-task learning model and capture the wisdom of the crowd, while the personalization layers can act as the task specific layers of the same multi-task learning model to capture user specific aspects.

In a separate line of work [**?** **?** ] motivated by scenarios in the healthcare domain, a federated transfer learning approach has been proposed for solving the personalization problem. A hidden assumption in this approach is that it is possible to separate the global training steps from the subsequent local personalization steps. We don't see any intuitive reason for such an assumption to hold for use cases outside of the healthcare domain and it definitely does not hold for the personalized image aesthetics and the personalized highlight detection tasks in **?** **?** ]. In the healthcare domain, separate organizations can be thought of as 'devices' that are disallowed from sharing raw data but could jointly train ML models via federated learning techniques [**?** ]. **?** ]

motivates various ways to split a federated deep learning model across organizational boundaries to satisfy different constraints on data sharing for vertically partitioned data. Although our 'base + personalization layers' construct is also based on a *splitting* principle, our specific split is different from those considered in the healthcare domain, and is motivated from and evaluated on completely different datasets and much smaller data volume distributions.

**Distributed Personalization and Recommendation:** Centralized algorithms for personalization and recommendation and their distributed counterparts have been extensively studied for various applications. The key assumption for many distributed ML algorithms is that data partitioning can be tightly controlled and communication delays are *i.i.d.* Remarkably, even asynchronous distributed algorithms like in [**?** ] can be shown to enjoy strong theoretical and empirical properties (like sublinear convergence for convex losses) under these assumptions. However, careful redesign and analysis of personalization and recommendation algorithms is needed for the federated setup owing to statistical and communication heterogeneity. Such efforts have appeared only recently in the form of **?** ] developing the first federated collaborative filtering algorithm and **?** ] formulating a federated meta-learning approach for recommendation. At present, these efforts disregard deep learning models, which is a significant shortcoming.

## 3 MODELING AND ALGORITHMIC SETUP

### 3.1 Model

Consider the setup shown in Figure 1 where all user devices share the same base layers and have unique personalization layers constituting the deep feed forward neural network models. We denote the number of base and personalized layers on each client by positive integers $K_B$ and $K_P$ respectively and let there be $N$ user devices. Let us designate the base layer weight matrices by $\boldsymbol{W}_{B,1}, \boldsymbol{W}_{B,2}, \ldots, \boldsymbol{W}_{B,K_B}$ and their corresponding vector-valued activation functions as $\boldsymbol{a}_{B,1}, \boldsymbol{a}_{B,2}, \ldots, \boldsymbol{a}_{B,K_B}$. While the base layer weight matrices may be of different dimensions, with a slight abuse of notation we will designate the tuple $\left(\boldsymbol{W}_{B,K_B}, \ldots, \boldsymbol{W}_{B,2}, \boldsymbol{W}_{B,1}\right)$ as $\boldsymbol{W}_B$, using the tensor notation. Analogously, for $j \in \{1, 2, \ldots, N\}$, we let the personalized layer weight matrices for the $j^{\text{th}}$ user device be designated as $\boldsymbol{W}_{P_j,1}, \boldsymbol{W}_{P_j,2}, \ldots, \boldsymbol{W}_{P_j,K_P}$, the corresponding vector-valued activation functions be denoted by $\boldsymbol{a}_{P_j,1}, \boldsymbol{a}_{P_j,2}, \ldots, \boldsymbol{a}_{P_j,K_P}$, and the tuple $\left(\boldsymbol{W}_{P_j,K_P}, \ldots, \boldsymbol{W}_{P_j,2}, \boldsymbol{W}_{P_j,1}\right)$ be notationally captured by $\boldsymbol{W}_{P_j}$. We will not use any non-trivial tensor calculus in the paper so the abuse of notation should not cause any confusion. Further, we will not explicitly track the dimensions of any weight matrices as the dimensions are irrelevant for the exposition in this paper. The dimensions of the domain and range of the vector valued activation functions are implicitly

unique so as to satisfy the forward pass/inference operation

$$\hat{y} = \boldsymbol{a}_{P_j,K_P}\Big(\boldsymbol{W}_{P_j,K_P}\cdots\boldsymbol{a}_{P_j,1}\Big(\boldsymbol{W}_{P_j,1}\boldsymbol{a}_{B,K_B}\big(\boldsymbol{W}_{B,K_B}\cdots\boldsymbol{a}_{B,1}(\boldsymbol{W}_{B,1}\boldsymbol{x})\cdots\big)\Big)\cdots\Big) \tag{1}$$

for any input data point $\boldsymbol{x}$ at the $j^{\text{th}}$ device. For brevity, we'll denote this forward pass operation at the $j^{\text{th}}$ device by $\hat{y} = f\Big(\boldsymbol{x}; \boldsymbol{W}_B, \boldsymbol{W}_{P_j}\Big)$. Thus, we implicitly assume that the weight tensor $\boldsymbol{W}_{P_j}$ captures all aspects of personalization at the $j^{\text{th}}$ device.

With a slight abuse of notation, we denote the joint distribution that generates (data, label) pairs at the $j^{\text{th}}$ device by $P_j$. Letting $l(\cdot,\cdot)$ denote the per sample loss function common to all devices, the learning goal in our setting is to minimize the average personalized population risk function

$$L^{PR}\big(\boldsymbol{W}_B, \boldsymbol{W}_{P_1}, \ldots, \boldsymbol{W}_{P_N}\big) = \frac{1}{N}\sum_{j=1}^{N}\mathbb{E}_{(\mathbf{x},\mathrm{y})\sim P_j}\Big[l\Big(\mathrm{y}, f\Big(\mathbf{x}; \boldsymbol{W}_B, \boldsymbol{W}_{P_j}\Big)\Big)\Big] \tag{2}$$

with respect to the weight tensors $\boldsymbol{W}_B, \boldsymbol{W}_{P_1}, \ldots, \boldsymbol{W}_{P_N}$. Since the true data generating distributions $P_j$ are unknown during training, we'll use the empirical risk at the $j^{\text{th}}$ device

$$L_j^{ER}\big(\boldsymbol{W}_B, \boldsymbol{W}_P\big) \triangleq \frac{1}{n_j}\sum_{i=1}^{n_j} l\big(y_{j,i}, f\big(\boldsymbol{x}_{j,i}; \boldsymbol{W}_B, \boldsymbol{W}_P\big)\big) \tag{3}$$

as a proxy for the corresponding population risk $\mathbb{E}_{(\mathbf{x},\mathrm{y})\sim P_j}[l(\mathrm{y}, f(\mathbf{x}; \boldsymbol{W}_B, \boldsymbol{W}_P))]$, where $n_j$ is the number of training samples available at the $j^{\text{th}}$ device and $\big(\boldsymbol{x}_{j,i}, y_{j,i}\big)$ is the $i^{\text{th}}$ such sample for $i \in \{1, \ldots, n_j\}$.

### 3.2 Algorithm

We will use stochastic gradient descent (SGD) as a subroutine in our proposed algorithm. Standard formulations of SGD [**?** ] for minimizing an empirical risk function like equation 3 requires us to specify the following: (a) the decision variables that would be updated by SGD and their initial values, (b) the batch size for partitioning the dataset $\big\{\big(\boldsymbol{x}_{j,i}, y_{j,i}\big) \;\big|\; 1 \le i \le n_j\big\}$ and the number of epochs over that dataset, and (c) the learning rate. We will denote by $\mathrm{SGD}\big(L_j^{ER}, \widehat{\boldsymbol{W}}_B, \widehat{\boldsymbol{W}}_P, \eta, e, b\big)$, the updated values of decision variable $\big(\boldsymbol{W}_B, \boldsymbol{W}_P\big)$, when the SGD is started from $\big(\boldsymbol{W}_B, \boldsymbol{W}_P\big) = \big(\widehat{\boldsymbol{W}}_B, \widehat{\boldsymbol{W}}_P\big)$ *w.r.t.* the loss function $L_j^{ER}\big(\boldsymbol{W}_B, \boldsymbol{W}_P\big)$ and executed with learning rate $\eta$ and batch size $b$ for $e$ epochs over the dataset. We have assumed that the random permutation of the dataset for an epoch is defined independent of the proposed algorithm.

Our proposed personalized federated training algorithm FedPer is described via Algorithms 1 and 2. The steps for the server component of FedPer are detailed under Algorithm 2 and the steps for the $j^{\text{th}}$ client are described under Algorithm 1. Intuitively, the server employs a FedAvg based approach to train the base layers globally whereas each client updates its base and personalized layers locally (between successive global aggregations) using a SGD style algorithm. To keep

**Algorithm 1** FedPer-Client($j$)

**Require:** $f(\cdot;\cdot,\cdot), e, b, \left\{\left(\boldsymbol{x}_{j,i}, y_{j,i}\right) \;\middle|\; i \in \left\{1,\ldots,n_j\right\}\right\}$
**Require:** $\eta_j^{(k)}$ for $k \in \mathbb{Z}_+$
1: Initialize $\boldsymbol{W}_{P_j}^{(0)}$ at random
2: Send $n_j$ to server
3: **for** $k = 1, 2, \ldots$ **do**
4: Receive $\boldsymbol{W}_B^{(k-1)}$ from server
5: $\left(\boldsymbol{W}_{B,j}^{(k)}, \boldsymbol{W}_{P_j}^{(k)}\right) \leftarrow \mathrm{SGD}_j\left(\boldsymbol{W}_B^{(k-1)}, \boldsymbol{W}_{P_j}^{(k-1)}, \eta_j^{(k)}\right)$
6: Send $\boldsymbol{W}_{B,j}^{(k)}$ to server
7: **end for**

**Algorithm 2** FedPer-Server

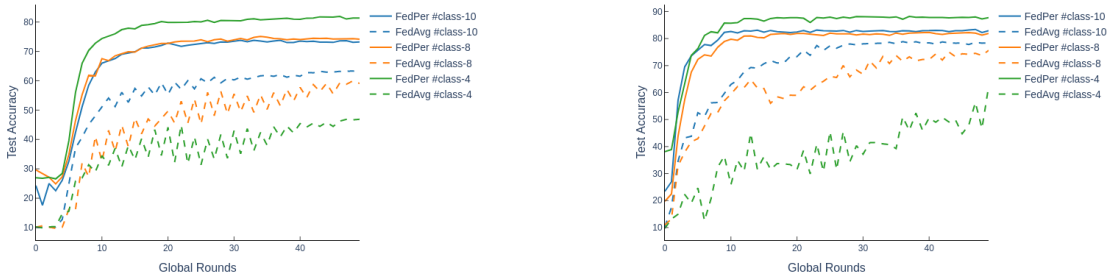
1: Initialize $\boldsymbol{W}_B^{(0)}$ at random
2: Receive $n_j$ from each client $j \in \{1,\ldots,N\}$ and compute $\gamma_j = n_j / \sum_{j=1}^N n_j$
3: Send $\boldsymbol{W}_B^{(0)}$ to each client
4: **for** $k = 1, 2, \ldots$ **do**
5: Receive $\boldsymbol{W}_{B,j}^{(k)}$ from each client $j \in \{1,\ldots,N\}$
6: Aggregate $\boldsymbol{W}_B^{(k)} \leftarrow \sum_{j=1}^N \gamma_j \boldsymbol{W}_{B,j}^{(k)}$
7: Send $\boldsymbol{W}_B^{(k)}$ to each client
8: **end for**

the exposition simple and relevant to the personalization driven statistical heterogeneity, we make the following assumptions that can be readily relaxed in implementation: (a) the dataset at any client doesn't change across global aggregations, (b) the batch size $b$ and the #epochs $e$ are invariant across clients and across global aggregations, (c) each client uses SGD to update $\left(\boldsymbol{W}_B, \boldsymbol{W}_{P_j}\right)$ between global aggregations, and (d) all $N$ user devices are active throughout the training process.

We denote the values assumed during the $k^{\text{th}}$ iteration by $(k)$ in the superscript. Between the $k^{\text{th}}$ and $(k+1)^{\text{th}}$ global aggregations, the $j^{\text{th}}$ device execute SGD with learning rate $\eta_j^{(k)}$ and let the updated value $\mathrm{SGD}\left(L_j^{ER}, \widehat{\boldsymbol{W}}_B, \widehat{\boldsymbol{W}}_P, \eta_j^{(k)}, e, b\right)$ be denoted by the shorthand $\mathrm{SGD}_j\left(\widehat{\boldsymbol{W}}_B, \widehat{\boldsymbol{W}}_P, \eta_j^{(k)}\right)$. During the $k^{\text{th}}$ global aggregation, the server computes a weighted combination of $\boldsymbol{W}_{B,j}^{(k)}$ across all clients $j \in \{1,\ldots,N\}$ using weights $\gamma_j = n_j / \sum_{j=1}^N n_j$ to arrive at $\boldsymbol{W}_B^{(k)}$.

## 4 EXPERIMENTS

We evaluate the performance of FedPer across multiple datasets and deep learning model families and contrast it with the behavior of FedAvg. We note that FedPer reduces to FedAvg when




(a) MobileNet-v1 (b) ResNet-34

Fig. 2. Performance of FedAvg vs FedPer on non-identical CIFAR-10 partition ($k \in \{4, 8, 10\}$)

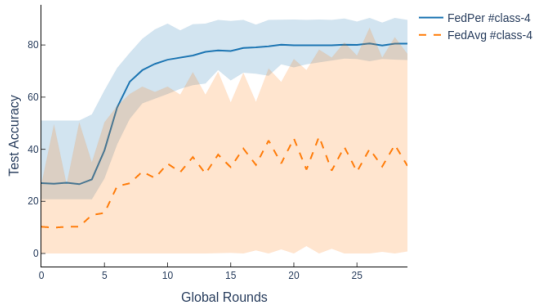


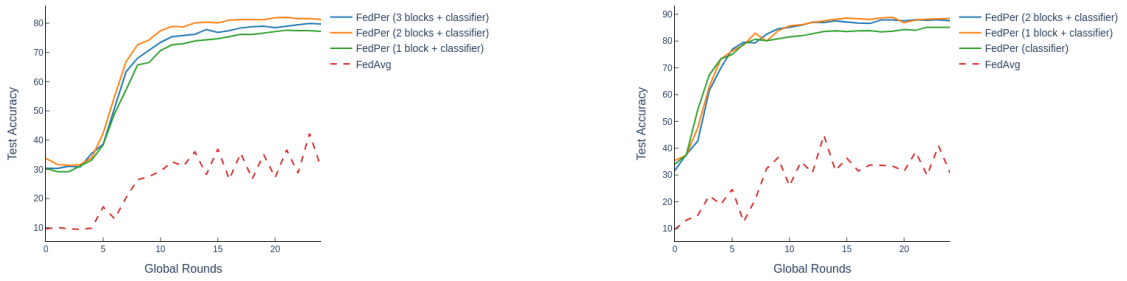

Fig. 3. Variation in MobileNet-v1 performance across clients for FedAvg vs FedPer on non-identical CIFAR-10 partition ($k = 4$)

$K_P = 0$, *i.e.* personalization layers are absent. Sections 4.2 and 4.3 respectively present the effect of statistical heterogeneity and changing $K_P$ for experiments with CIFAR datasets. Corresponding results on the FLICKR-AES dataset are presented in 4.5. In the spirit of reproducibility, we have made all datasets, implementations for experiments, and results available at https://bit.ly/35dKebE.

### 4.1 Datasets, Model Architectures, and Implementation Details

**Datasets:**

(1) CIFAR-10 and CIFAR-100 are image classification datasets with 10 and 100 labeled classes respectively [**?** ]. Both have 50,000 training samples and 10,000 testing samples. Each image has a unique label.




(a) MobileNet-v1 (b) ResNet-34

Fig. 4. Performance of FedPer on non-identical CIFAR-100 partition *w.r.t.* #personalization layers

(2) FLICKR-AES dataset was compiled by **?** ] to study personalized image aesthetics. It has 40,000 photographs from www.flickr.com with aesthetic ratings (between 1 and 5) collected via Amazon Mechanical Turk. Each image is rated by 5 different users and there are a total of 210 users. Each user has rated between 60 and 290 images with average being 160 ratings. We randomly split the data of each user into an 80% training set and a 20% testing set.

**Model Architectures:** We work with the ResNet-34 and MobileNet-v1 families of convolution neural networks. While ResNet model family achieves state-of-the-art performance for many image classification tasks, MobileNet is an efficient model family for mobile vision applications with low compute resources. Both of these families are constructed by composing their respective basic block multiple times. ResNet-34 and MobileNet-v1 have 16 and 11 basic blocks respectively. ResNet's basic block is a residual block consisting of 2 convlayers and a residual connection, while MobileNet's basic block consists of a depthwise and a pointwise convolutional layer. We will delineate base and personalization layers in units of basic blocks.

**Implementation Details:** We run each experiment for 100 global aggregations, with $e = 4$ epochs for SGD between successive global aggregations. We use constant learning rate $\eta_j^{(k)} = 0.01$ across global aggregations and across clients.

- Experiments with FLICKR-AES dataset use a random subset of $N = 30$ users as clients, with SGD batch size $b = 4$. This dataset is naturally unbalanced in terms of $n_j$ and statistically heterogeneous in terms of $P_j$.
- Experiments with CIFAR-10 and CIFAR-100 use $N = 10$ clients, with SGD batch size $b = 128$. Data volumes are balanced across clients in terms of $n_j$, but are non-identically partitioned in terms of $P_j$ by restricting the samples of any particular client to belong to atmost $k$ classes. The

number of samples from a particular class is distributed equally across clients allowed to sample from it, and the level of statistical heterogeneity is varied by changing the value of $k$.

### 4.2 Effect of Statistical Heterogeneity

Figure 2 presents the test set accuracies for FedAvg vs FedPer (averaged across clients) on ResNet-34 and MobileNet-v1 as a function of global aggregation rounds for different levels of statistical heterogeneity ($k \in \{4, 8, 10\}$). $k = 4$ corresponds to a highly non-identical data partition, whereas $k = 10$ corresponds to an identical data partition across the clients. Figure 3 shows the variation of the test set accuracies across clients (via the shaded regions) for FedAvg vs FedPer for $k = 4$ with MobileNet-v1. For results in both of these figures, the classifier layer (final fully connected layer) and the immediately preceding basic block were used as personalization layers for FedPer.

We observe from Figure 2 that FedPer is significantly better both in terms of convergence speed *w.r.t.* global rounds and the client averaged test accuracy achieved at steady state. Interestingly, as the data partition becomes more identical (parameterized by $k$ moving from 4 to 10), FedPer's performance comes closer to that of FedAvg. The qualitative nature of these results remains unchanged even if CIFAR-10 is replaced by CIFAR-100 and we select $k \in \{40, 80, 100\}$ (see supplementary material for plots).

Figure 3 shows that FedPer also results in lower variation in test accuracies across clients. This is important for fairness in learning. The large cross-client variation in test accuracies for FedAvg is somewhat surprising. However, the magnitude of this variation does decrease when non-identicality of the data partition is reduced by increasing $k$ from 4 to 10 (see supplementary material for plots).

### 4.3 Effect of Personalization Layers

We consider personalization layers to include the classifier layer (final fully connected layer) and last few basic blocks. Since we delineate personalization layers in units of basic blocks, with a slight abuse of notation, we can use $K_P$ to denote the number of basic blocks included in the personalization layers. $K_P = 4$ refers to the classifier layer and the last 3 basic blocks included in the set of personalization layers, whereas $K_P = 1$ refers to only the classifier layer included in the set of personalization layers.

Figure 4 plots test accuracies (averaged across clients) *w.r.t.* CIFAR-10 for FedPer on ResNet-34 and MobileNet-v1 as a function of global aggregation rounds for different number of personalization layers ($K_P \in \{1, 2, 3, 4\}$). Interestingly, there doesn't seem to be a clear correlation between $K_P$ and the client averaged test accuracy at steady state, except that having atleast one personalization

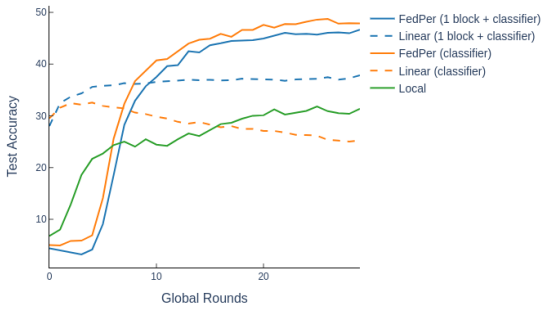



Fig. 5. Effect on the performance of MobileNet-v1 on CIFAR-100 before and after replacing the base layers with linear layers

layer helps. For both MobileNet-v1 and ResNet-34, $K_P = 2$ seems to achieve the best performance on CIFAR-10, but $K_P = 1$ seems to be the best for CIFAR-100.

### 4.4 Do Base Layers Learn Anything?

In theory, it is possible that the personalization layers may be sufficient for the learning tasks at hand rendering the base layers redundant. To test whether this is the case, we replace the base layers at each client with a linear fully connected layer and train with FedPer. If the base layers are redundant, then there should not be a significant drop in test accuracy (as compared to ResNet-34 or MobileNet-v1) when this model is trained. Figures 5 and 6 confirm that this is not the case, atleast for CIFAR-100. In fact, the performance of purely local training confirms that the number of training samples per client is not large enough to learn individual client models to high degrees of accuracy.

### 4.5 Performance on FLICKR-AES

Figure 7 and 8 shows the test performance of the MobileNet-v1 and ResNet-34 models on the FLICKR-AES dataset. It can be seen from the results that FedAvg is unfit for modeling personalization tasks. The Performance of the MobileNet model trained using FedAvg is quantitatively very different when compared to its performance on CIFAR datasets and is similar to random guessing. This is because, the FedAvg approach has no way of capturing the personal preferences of the user as it has the same copy of the model on all clients.

Models trained using our FedPer approach captures the user preferences through the personalization layers. While the models used here are not state-of-the-art for this task, it provides sufficient evidence to show that our FedPer approach has the ability to model personalized tasks. Results of

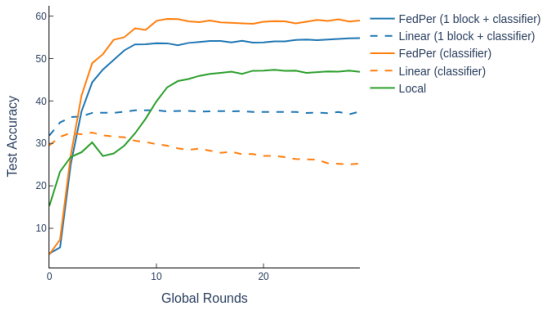



Fig. 6. Effect on the performance of ResNet-34 on CIFAR-100 before and after replacing the base layers with linear layers

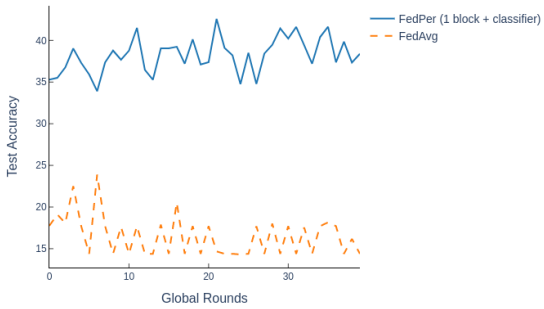



Fig. 7. Performance of MobileNet-v1 on FLICKR-AES when trained using FedAvg and FedPer

the experiment where we replace the base layers of the model with linear layer for MobileNet-v1 is given in Figure 9. Similar to the CIFAR-100 dataset, we see a drop in the performance of morphed models thereby again confirming our hypothesis that base layers capture complex image features through federated training.

## 5 CONCLUSION

In this work, we propose FedPer a novel approach for capturing the personalization aspect of users in the federated learning setup. FedPer achieves this by viewing the deep learning models as base + personalization layers. While the base layers are trained in a collaborative manner using the existing federated learning approach, the personalization layers are trained locally thereby enabling our approach to capture personalization aspects of users. Results indicate that having

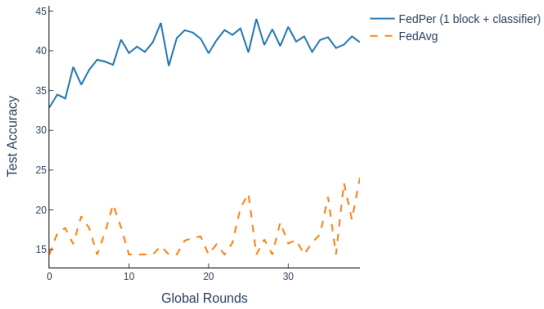



Fig. 8. Performance of ResNet-34 on FLICKR-AES when trained using FedAvg and FedPer

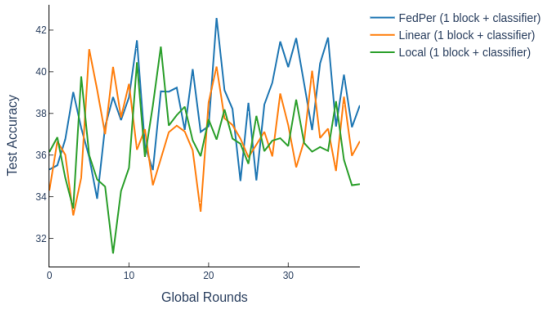



Fig. 9. Effect on the performance of MobileNet-v1 on FLICKR-AES before and after replacing the base layers with linear layers

base + personalization layers help in combating the ill-effects of statistical heterogeneity. Empirical results on FLICKR-AES and CIFAR datasets demonstrates the ineffectiveness of FedAvg and the effectiveness of FedPer in modelling the personalization tasks.

## A APPENDIX

### A.1 Effect of Statistical Heterogeneity

Figures 11, 12 are plots on CIFAR-100 corresponding to CIFAR-10 plots in Appendix 4.2 and Figure 13 is the cross-client variation in test accuracies with identical data partitioning at $k = 10$.

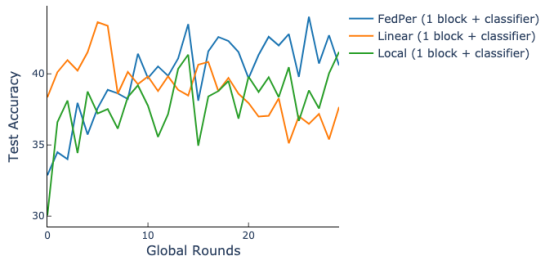



Fig. 10. Effect on the performance of ResNet-34 on FLICKR-AES before and after replacing the base layers with linear layers

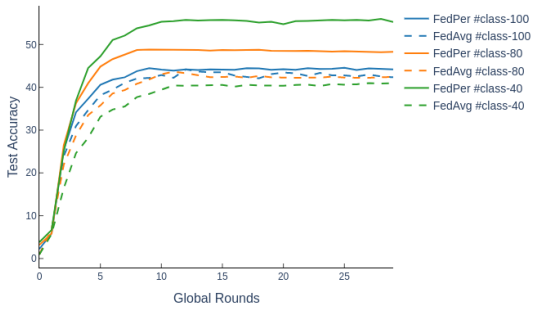



Fig. 11. Effect of Non-IID nature on the performance of ResNet-34 on CIFAR-100 when trained using FedAvg and FedPer

## A.2 Effect of Personalization Layers

Figure 14 shows that for CIFAR-100, $K_P = 1$ seems to do the best in terms of test accuracies.

## A.3 Effect of FineTuning

In the current FedPer approach, each global round involves the server sending the base layer parameters $\boldsymbol{W}_B^{(k-1)}$ to the clients, clients updating their local model parameters by performing e local epochs and then sending the updated base layer parameters $\boldsymbol{W}_{B,j}^{(k)}$ to the server. The server then aggregates the base layer parameters to get the updated $\boldsymbol{W}_B^{(k)}$.

In this experiment, at the start of each global round, we fine tune the personalization layer parameters of the clients for 1 epoch by freezing the base layers. We hypothesize that fine tuning

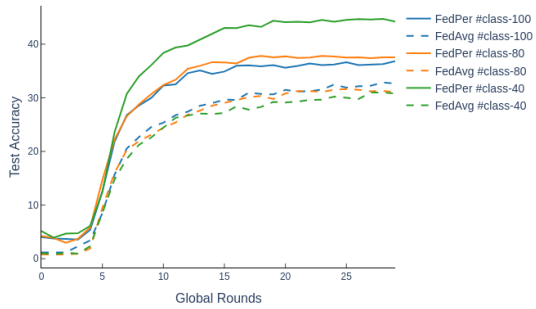



Fig. 12. Effect of Non-IID nature on the performance of MobileNet-v1 on CIFAR-100 when trained using FedAvg and FedPer

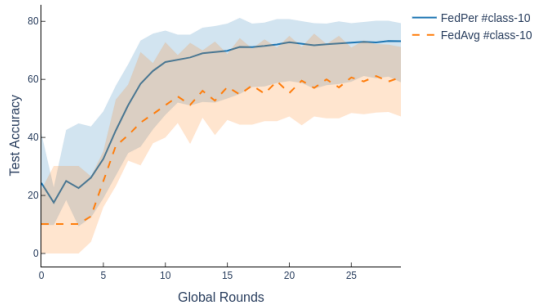
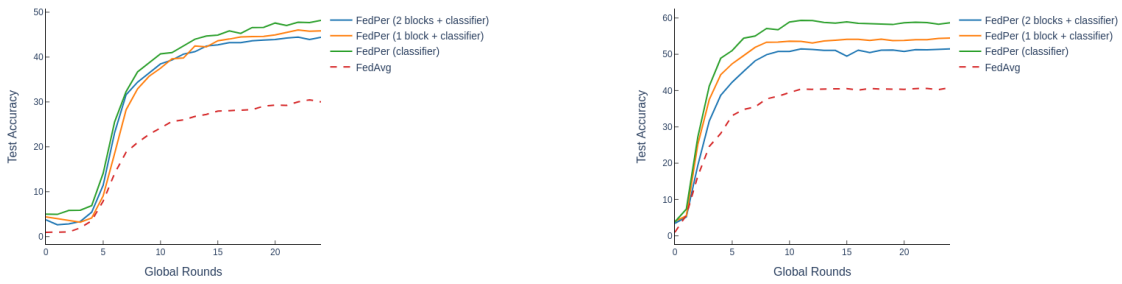



Fig. 13. Variation in the client models performance when training MobileNet-v1 on CIFAR-10 using FedAvg and FedPer

the personalization layers after receiving the base layer parameters from the server will help the local client models to accommodate to the previous round changes in the base layer parameters better. Results of fine-tuning of MobileNet-v1 and ResNet-34 on CIFAR-100 dataset are shown in Figures 15 and 16. It can be seen from the results that fine-tuning has a positive effect on the client models by improving their accuracy thereby confirming our hypothesis.

FLICKR-AES: The effect of fine-tuning on the performance of MobileNet-v1 and ResNet-34 are given in Figure 17 and 18. Interestingly fine-tuning of personalization layers does not have any noticeable effect in the performance of the client models.




(a) MobileNet-v1 (b) ResNet-34

Fig. 14. Performance of FedPer on non-identical CIFAR-100 partition *w.r.t.* #personalization layers

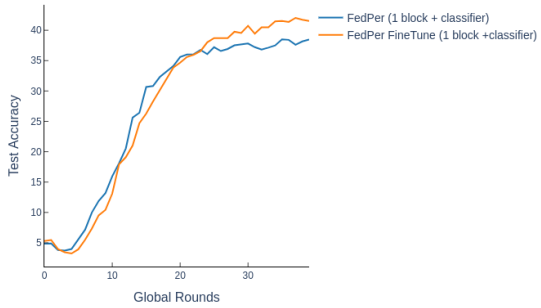



Fig. 15. Effect on the performance of MobileNet-v1 on CIFAR-100 with/without fine-tuning personalization layers

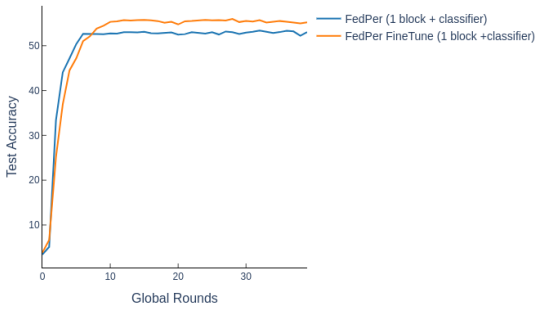



Fig. 16. Effect on the performance of ResNet-34 on CIFAR-100 with/without fine-tuning personalization layers

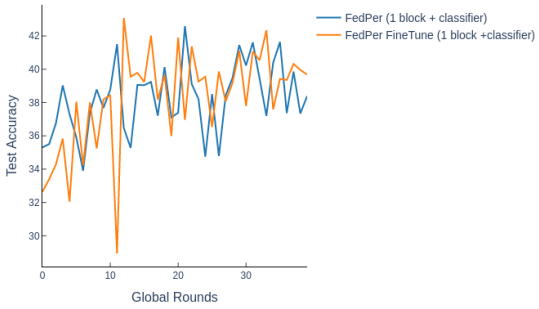



Fig. 17. Effect on the performance of MobileNet-v1 on FLICKR-AES with/without fine-tuning personalization layers

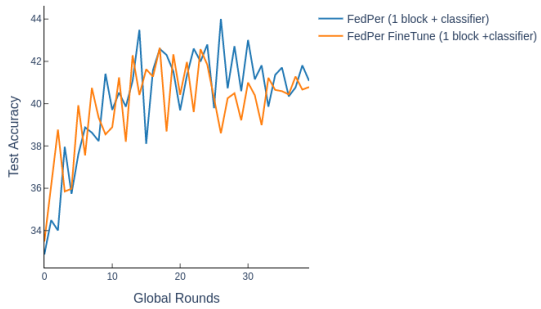



Fig. 18. Effect on the performance of ResNet-34 on FLICKR-AES with/without fine-tuning personalization layers