ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095262

EFFICIENT PERSONALIZED FEDERATED LEARNING ON SELECTIVE MODEL TRAINING

Yeting Guo[*], *Fang Liu*[†], *Tongqing Zhou*[*], *Zhiping Cai*[*], *Nong Xiao*[*]

[*]College of Computer, National University of Defense Technology
[†]School of Design, Hunan University

ABSTRACT

Personalized Federated Learning (FL) handles the data heterogeneous problem by tailoring local models for each distributed data owner. Previous studies first train a highly-adaptable global model and then transfer it for personalization. However, the additional training aggravates burden of resource-limited end devices. Training a personalized local sub-network is a promising efficient solution. It normally prunes the global model by parameters' scalar magnitude. In this paper, we found that the vector magnitude, i.e. the parameter stability, could further promote personalized FL. Driven by the local data characteristics, the values of some model parameters are hardly changed in their updates. But they consume the same resources as the changed ones. Thus, we propose Star-PFL, a STability-AwaRe algorithm for efficient FL Personalization. In Star-PFL, the data owner focuses on training non-stabilized parameters, and decreases the resource wastes on stabilized ones. Experimental results on two real-world biomedical datasets demonstrate that Star-PFL improves the accuracy (3.1%↑) and decreases the resource costs (communication 36.3%↓, computation 18.3%↓) than 5 typical baselines. The code is available at https://github.com/Guoyeting/Star-PFL.

Index Terms— federated learning, deep learning, data privacy

1. INTRODUCTION

Pervasive end devices have facilitated the ever-present and ubiquitous data collection in our daily life. It nourishes various intelligent applications, but compromises user privacy.

Federated Learning (FL) provides a privacy preserving solution for distributed data owners to collaboratively learn a machine learning model [1, 2]. Specifically, FL allows data owners (referred to as *client*) keep sensitive data at local and share their local trained models to the cloud server to aggregate a global model. It has attracted widespread interest in fields such as medical diagnosis and voice assistant [3, 4, 5].

Data between clients usually have different characteristics, such as demographics and behavior preferences. In such data heterogeneity setting, the unified global model, generated by directly averaging all the distributed models, can not meet the actual needs of individuals well [6, 7]. Thus, we target at developing an efficient personalized FL system.

Extensive research regarding personalized FL has been proposed [8, 9]. Some studies have applied meta learning [10, 11] and transfer learning [12, 13] for personalization. They usually take two steps: the clients firstly train a global model collaboratively, and then train personalized models with local data based on the global model. But it increases the computation cost. Different with the above **additional** training, some studies have claimed that **pruning** can also achieve the personalization but more efficiently [14, 15]. They prune the global model and directly share the pruned model during training. The pruning is mostly based on parameters' **scalar** magnitude. Little attention has been paid to the impact of **vector** magnitude, i.e. the fluctuation of parameter updates, on the personalization.

The fluctuation of parameter updates implies the data characteristics. It's observed that some parameters can reach stability in early training, while some insist non-stabilized. The stabilized ones hardly changes their values, but still consume the same resources as the non-stabilized ones. Thus, this paper seeks to explore whether the end device can achieve efficient personalization by reducing the wasted effort in training already stabilized parameters and focusing on training the non-stabilized ones. However, it's still technically challenging: 1) how to distinguish the non-stabilized parameters in FL for efficient personalization? 2) how to guarantee the model convergence with such selective partial training?

Given these challenges, we propose *Star-PFL*, a STability-AwaRe algorithm for efficient FL Personalization. Firstly, we analyze the parameter stability on both sides of the global server and clients. The global server analyzes the stability of global model parameters and inform the clients to only upload the updates to the global non-stabilized parameters. Clients discriminate local non-stabilized parameters, and show their personalized talent for model optimization via only training these parameters. In that way, the resource efficiency is improved. Second, we adaptively check the parameter stability to guarantee the model convergence. It's resource consuming to analyze the stability in each aggregation round. Thus, we

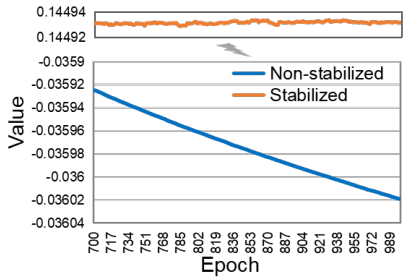

Fig. 1. Fluctuation of two randomly selected parameters

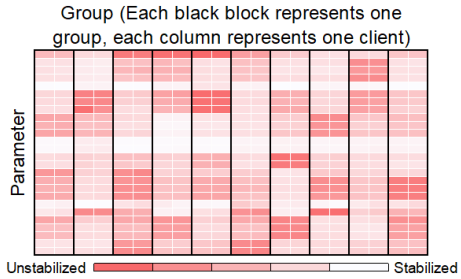

Fig. 2. Parameter stability among clients

check periodically and adjust the check interval based on the feedback. It effectively avoid the model performance degradation caused by some temporarily stabilized parameters.

2. MOTIVATION

We conducted some test bed measurements on the fluctuation of the local parameter updates in FL settings. Here we train the dataset OrganMNIST_Sagital [16] using LeNet-5 [17]. The model aggregation follows the classical FedAvg [18].

Fig. 1 shows the value fluctuation of two randomly selected local model parameters during training. It's observed that although the absolute value of the parameter marked orange is far greater than that of the one marked blue, the former has been stabilized with barely changed value, while the latter remains non-stabilized with constantly decreased value. It motivates us to *leverage the stability to improve the resource efficiency instead of purely referring to the absolute value*.

Fig. 2 illustrates the stability of some parameters among clients. Stability is defined as the ratio of the absolute value of the sum of the recent updates and the sum of the absolute value of updates [19]. Note that we target at optimizing both the personalization and resource cost simultaneously rather than purely reducing communication cost discussed in [19]. In Fig. 2, the clients in the same group have the data with the same label. It's found that the parameter stability is different among clients who have different data characteristics. And clients with the similar characteristics have the similar stabilized parameters. It's implied that *the stability can reflect the data characteristics and promote the personalization*.

Table 1. Main notation descriptions in Star-PFL

Notation	Description
W	Model parameters
ΔW	Update of W
R	Set of consecutive updates ΔW
T	Threshold to determine whether W is stabilized
$\overline{W}$	Non-stabilized model parameters in W
$\Delta\overline{W}$	Update of $\overline{W}$
F	Frequency of checking parameter stability
M	Mask indicating whether W is non-stabilized
M'	Mask indicating whether W is just switched from stabilized to non-stabilized for the check
L	Length of time that parameters have been frozen

3. METHODOLOGY

Based on the above findings, we propose Star-PFL that leverages parameter stability for efficient personalized FL. The workflow is illustrated in Algorithm 1. The main notations are listed in Table 1. We use subscripts to distinguish these notations of the global model and the local model. For example, W_g denotes the global model parameters, and W_l is the local ones. The main designs are elaborated as follows.

3.1. Two-side Model Parameter Stability Measurement

A parameter is regarded *stabilized* if its value hardly changes in the recent updates. Algorithm 2 describes the parameter stability measurement. R records a set of consecutive parameter updates. T is a preset threshold. If one parameter's $stability$ is smaller than T, it's indicated that the parameter is stabilized and its mask value M is set 0, otherwise 1. To identify stabilized parameters in FL, we measure parameter stability on both the global server and the client.

Server: The global server randomly initial the global parameter W_g. In the initialization, R_g is null, $\{M_g, M'_g, L_g\}$ is set 0, and F_g is set 1, which indicates no stabilized global parameters in the beginning (line 2[1]). In each round, the global server firstly measures the stability of the parameters that are not yet stabilized, and determines M_g and the non-stabilized parameter $\overline{W}_g$ (line 3-7). Parameters whose M'_g value is 1 are just activated to non-stabilized for stability check. We would discuss their stability later. The global server broadcasts W_g and M_g, and receives clients' updates on non-stabilized parameters (line 8-10). It calculates the weighted average of these updates, and updates the non-stabilized $\overline{W}_g$ and R_g (line 11-12). R_g records the most recent global updates.

Client: The client i performs the similar initialization on $\{R_l, M_l, M'_l, F_l, L_l\}$ (line 18). In each round, the client firstly initials the local W_l^i to the global W_g. Then it measures the stability of local parameters that are not yet stabilized,

[1] All the line numbers refer to the line numbers in Algorithm 1.

Algorithm 1: Workflow of Star-PFL

Procedure Server
  Initialize W_g, R_g, M_g, M'_g, F_g, L_g;
  for *each round* $t = 1, 2, \dots$ **do**
    # Measure the stability of the global model
    $Para_g^{un} \leftarrow$ parameters where $M_g == 1$ and $M'_g == 0$;
    $M_g \leftarrow Measure(R_g, T_g)$ for $Para_g^{un}$;
    $\overline{W}_g \leftarrow W_g$ where $M_g == 1$;
    # Aggregate local models
    Send W_g, M_g to clients $\{1 \dots N\}$;
    Receive $\{\Delta\overline{W}_l^1 \dots \Delta\overline{W}_l^N\}$ from clients;
    $\Delta\overline{W}_g \leftarrow \sum_{i=1}^{N} \frac{|D_i|}{|D|}\Delta\overline{W}_l^i$;
    Update R_g with $\Delta\overline{W}_g$, $\overline{W}_g \leftarrow \overline{W}_g + \Delta\overline{W}_g$;
    # Check the stability of the global model
    $L_g \leftarrow L_g + 1$ where $M_g == 0$;
    $M'_g \leftarrow Check(M'_g, R_g, F_g, L_g, T_g)$;
    $M_g \leftarrow M'_g == 1?1 : M_g$;
Procedure Client
  Initialize R_l, M_l, M'_l, F_l, L_l;
  for *each round* $t = 1, 2, \dots$ **do**
    Initialize $W_l^i \leftarrow W_g$;
    # Measure the stability of the local model
    $Para_l^{un} \leftarrow$ parameters where $M_l == 1$ and $M'_l == 0$;
    $M_l \leftarrow Measure(R_l, T_l)$ for $Para_l^{un}$;
    $\overline{W}_l^i \leftarrow W_l^i$ where $M_l == 1$;
    # Train the personalized local model
    for *each epoch* $e = 1, 2, \dots$ **do**
      Train $\overline{W}_l^i$ with local dataset D_i;
      Update R_l with $\Delta\overline{W}_l^i$ in early epochs;
    $\overline{W}_g \leftarrow W_g$ where $M_g == 1$ and $M_l == 1$;
    $\Delta\overline{W}_l^i \leftarrow \overline{W}_l^i - \overline{W}_g$;
    Send $\Delta\overline{W}_l^i$ to the server;
    # Check the stability of the local model
    $L_l \leftarrow L_l + 1$ where $M_l == 0$;
    $M'_l \leftarrow Check(M'_l, R_l, F_l, L_l, T_l)$;
    $M_l \leftarrow M'_l == 1?1 : M_l$;

Algorithm 2: Function *Measure*

Input: $R = \{\Delta W^1, \Delta W^2 \dots\}, T$
Output: M
$stability \leftarrow \frac{|\sum_{j=0}^{|R|} \Delta W^j|}{\sum_{j=0}^{|R|} |\Delta W^j|}$;
$M \leftarrow stability > T?1 : 0$;

Algorithm 3: Function *Check*

Input: M',R,F,L,T
Output: M'
$M \leftarrow Measure(R, T)$ where $M' == 1$;
$F \leftarrow F + 1$ where $M' == 1$ and $M == 0$;
$F \leftarrow F/2$ where $M' == 1$ and $M == 1$;
Reinitialize M';
 # Unfreeze the parameters reaching the checkpoint
$M' \leftarrow 1, L \leftarrow 0$ where $F \leq L$;

identifies and trains the non-stabilized $\overline{W}_l^i$. Different with the global server, the client records the updates only in the early local training epochs in R_l (line 19-23) due to its limited resources. The length of R_l depends on its resource capacity.

3.2. Stability-based Personalized Model Training

Even though the stabilized parameters hardly changes, they still cost the same communication and computation resources as the non-stabilized ones. In Star-PFL, we propose to regard the local non-stabilized $\overline{W}_l^i$ as the personalized part of the model. The client trains $\overline{W}_l^i$ for personalization while keeping the stabilized parameters frozen. And it need not send the updates on the stabilized ones. Thus, both the computation and communication resource costs could be saved.

Specifically, during the local training, the client i performs forward propagation with the local W_l^i, and calculates the loss value. Based on the loss value, it performs backward propagation only to the local non-stabilized $\overline{W}_l^i$. In that way, $\overline{W}_l^i$ is optimized to better fit the local data while the local stabilized parameters still remain the initialized value. Thus, the client only need to inform the global server the updates on parameters which are non-stabilized in both the global and local model, that is, both M_g and M_l are 1 (line 27-28).

3.3. Adaptive Stability Check to Guarantee Convergence

Some parameters are stabilized temporarily. If they remain frozen without learning after being detected as stabilized, the model performance would be degraded. In view of this, we would check the stability for the model convergence.

In the check, we switch the state of some stabilized parameters to non-stabilized. The mask M' records the switched parameters. If switched, its M' value is set 1, otherwise 0. Then we observe their fluctuation when they return to the training process. Frequent switching brings high resource costs and oscillatory noises to model optimization. Thus, we adaptively adjust the check frequency F based on their fluctuation. If they are still stabilized during this check, F would be additively increased, otherwise multiplicatively reduced. L records the length of time that parameters have been frozen. When L reaches F, the switch would be triggered.

The procedure is shown in Algorithm 3. And it's embedded at the end of each training round in FL (line 13-16, 32-35).

4. EVALUATION

4.1. Experiment Setup

Datasets. We evaluate the performance on two biomedical datasets: OrganMNIST_Axial and OrganMNIST_Sagital (Axial, Sagital for short) [16]. Both the two datasets contain abdominal CT images of 11 classes. The number of their images is 58,850 and 25,221. The images are distributed among 20 clients following a Dirichlet distribution Dir(α) to simulate the heterogeneous settings. Each client uses 60%, 20% and 20% data for training, validation and testing, respectively.
Model settings. We leverage LeNet-5 to train the biomedical images [17]. It consists of two convolutional layers, two batch normalization layers, two MaxPool layers and three dense layers. The client uploads the local model to the server every 10 local training epochs. The length of record R_g and R_l is 10 and 5. The threshold T_g and T_l are set 0.1 and 0.1.
Baselines. We compare Star-PFL with the following baselines. 1) **Local** is the basic baseline in which each client independently trains personalized models. 2) **FedAvg** [18] is a classical FL baseline that all the clients use a unified global model. 3) **FedBN** [9] achieves the personalization via keeping the local batch normalization without being affected by the global server. 4) **FedAP** [20] transfers the global model locally with adaptive batch normalization. 5) **LotteryFL** [14] prunes the global model for personalization by the model parameters' scalar magnitude.

4.2. Results

The evaluation aims to answer the two questions: whether the personalized models work well on local data; whether training these models is resource friendly to the end devices.
Test accuracy of the personalized model. We measure the average classification accuracy on each client's test dataset. The results are shown in Table 2. We simulate different levels of data heterogeneity by setting the distribution factor α to 1, 0.1 and 0.01. The larger α, the more unbalanced the data distribution among the clients. The results illustrate that our Star-PFL always has the best or the second best test accuracy.
Communication cost and local training cost. Table 3 summarizes the resource cost of baselines. Specifically, *Comm* refers to the average communication cost of each client in each round. *FLOPs* is the number of operands to be executed for local training in each round. *Round* is the number of rounds required for model convergence. From the result, it's found that Star-PFL has the least communication and local computation cost in each round among FL-based baselines. Besides, it converges faster than personalized FL baseline FedBN and FedAP. Although slower than LotteryFL, Star-PFL achieves higher test accuracy than LotteryFL.

Table 2. Test accuracy under different data heterogeneities

Task	Methods	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$
Axial	Local	0.8892	0.8832	0.8324
	FedAvg	0.9072	0.9106	**0.9181**
	FedBN	0.9162	**0.9269**	0.9112
	FedAP	0.9170	0.9225	0.9086
	LotteryFL	0.8993	0.9065	0.8884
	Star-PFL	**0.9257**	0.9247	0.9134
Sagital	Local	0.7495	0.7290	0.6383
	FedAvg	0.6993	0.7144	0.7258
	FedBN	0.7548	0.7504	0.7200
	FedAP	0.7852	0.7786	0.7333
	LotteryFL	0.7816	0.7887	0.7468
	Star-PFL	**0.8043**	**0.7998**	**0.7470**

Table 3. Resource consumption of baselines ($\alpha = 0.1$)

Task	Methods	Comm (MB)	FLOPs (1e10)	Round
Axial	Local	/	3.005	141
	FedAvg	3.403	3.005	286
	FedBN	3.396	3.005	389
	FedAP	3.403	3.005	233
	LotteryFL	2.315	2.973	**98**
	Star-PFL	**2.249**	**2.912**	174
Sagital	Local	/	3.005	**42**
	FedAvg	3.403	3.005	94
	FedBN	3.396	3.005	369
	FedAP	3.403	3.005	283
	LotteryFL	2.322	2.948	100
	Star-PFL	**2.320**	**2.937**	163

5. CONCLUSION

Different with the common pruning based on parameters' scalar magnitude, we highlight the importance of vector magnitude to personalized FL. The stability of parameter updates implies the local data characteristics. In our Star-PFL, the client only optimize the non-stabilized parameters for personalization, and freeze the already stabilized ones for resource saving. Experimental results demonstrate its efficacy.

Acknowledgement

This work is supported by National Key Research and Development Program of China (2022YFF1203001), National Natural Science Foundation of China (62172155, 61832020, 62072465, 62102425), Natural Science Foundation of Guangdong Province (2018B030312002), and Science and Technology Innovation Program of Hunan Province (2021RC2071, 2022RC3061). The corresponding author is Fang Liu.

6. REFERENCES

[1] Jie Ding, Eric Tramel, Anit Kumar Sahu, Shuang Wu, Salman Avestimehr, and Tao Zhang, "Federated learning challenges and opportunities: An outlook," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 8752–8756, IEEE.

[2] Abhishek Kumar, Vivek Khimani, Dimitris Chatzopoulos, and Pan Hui, "Fedclean: A defense mechanism against parameter poisoning attacks in federated learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 4333–4337, IEEE.

[3] Suhas BN and Saeed Abdullah, "Privacy sensitive speech analysis using federated learning to assess depression," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 6272–6276, IEEE.

[4] Yeting Guo, Fang Liu, Zhiping Cai, Li Chen, and Nong Xiao, "FEEL: A federated edge learning system for efficient and privacy-preserving mobile healthcare," in *International Conference on Parallel Processing (ICPP)*. 2020, pp. 9:1–9:11, ACM.

[5] Junqiao Fan, Xuehe Wang, Yanxiang Guo, Xiping Hu, and Bin Hu, "Federated learning driven secure internet of medical things," *IEEE Wirel. Commun.*, vol. 29, no. 2, pp. 68–75, 2022.

[6] Junjie Pang, Yan Huang, Zhenzhen Xie, Qilong Han, and Zhipeng Cai, "Realizing the heterogeneity: A self-organized federated learning framework for iot," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3088–3098, 2021.

[7] Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *The Web Conference (WWW)*, 2021, pp. 935–946.

[8] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant, "Survey of personalization techniques for federated learning," *CoRR*, vol. abs/2003.08673, 2020.

[9] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," in *International Conference on Learning Representations (ICLR)*, 2021.

[10] Ziqiu Chi, Zhe Wang, and Wenli Du, "Heterogeneous federated meta-learning with mutually constrained propagation," *IEEE Intell. Syst.*, vol. 37, no. 2, pp. 44–54, 2022.

[11] Zhengyang Ai, Guangjun Wu, Xin Wan, Zisen Qi, and Yong Wang, "Towards better personalization: A meta-learning approach for federated recommender systems," in *Knowledge Science, Engineering and Management (KSEM)*. 2022, vol. 13369, pp. 520–533, Springer.

[12] Siwei Feng, Boyang Li, Han Yu, Yang Liu, and Qiang Yang, "Semi-supervised federated heterogeneous transfer learning," *Knowl. Based Syst.*, vol. 252, pp. 109384, 2022.

[13] Cen Chen, Tiandi Ye, Li Wang, and Ming Gao, "Learning to generalize in heterogeneous federated networks," in *ACM International Conference on Information & Knowledge Management*. 2022, pp. 159–168, ACM.

[14] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li, "Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning," in *IEEE/ACM Symposium on Edge Computing (SEC)*. 2021, pp. 68–79, IEEE.

[15] Saeed Vahidian, Mahdi Morafah, and Bill Lin, "Personalized federated learning by structured and unstructured pruning under data heterogeneity," in *IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*. 2021, pp. 27–34, IEEE.

[16] Jiancheng Yang, R Shi, and Bingbing Ni, "Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis," *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 191–195, 2021.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*. 2017, vol. 54, pp. 1273–1282, PMLR.

[19] Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang, "Communication-efficient federated learning with adaptive parameter freezing," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*. 2021, pp. 1–11, IEEE.

[20] Yiqiang Chen, Wang Lu, Jindong Wang, Xin Qin, and Tao Qin, "Federated learning with adaptive batchnorm for personalized healthcare," *CoRR*, vol. abs/2112.00734, 2021.