Meta-learning and Personalization layer in Federated learning

Bao-Long Nguyen[1][0000−0002−6411−8943]
and Tat-Cuong Cao[1][0000−0003−1803−843X]

VNU HCM University of Science, Ho Chi Minh City 749000, Vietnam

Abstract. The concept of *Federated Learning* (FL) and the algorithm *Federated Averaging* were born, which is seen as an alternative to centralized training. This solution not only achieves almost the same efficiency as existing deep learning methods, solves the hardware cost problem, but also ensures the privacy of user data. However, in the face of highly individualized and heterogeneous data per user (not independent and identically distributed or non-IID data), the system suffered severe performance degradation [22]. This study improves the performance and personalization of the FL system on non-IID data by the algorithm `FedMeta-Per` - a combination of the meta-learning training method [9] and the personalization layer technique [23] into the FL system.

Keywords: Federated learning · non-IID · Meta-learning · Personalization layer.

1 Introduction

In the era of Internet of Things (IoT), facing a huge amount of data at the edge devices generated every second, the centralized training process presents many disadvantages. First, the data at the edge devices often contains sensitive information about the user. When it has to be transmitted to a server for training, this information can be exposed, seriously affecting data privacy. Second, the cost of transmitting information between the server and the clients is very expensive because a large amount of data must be transferred to the server. Third, the server must have large computing power and storage to conduct the training process.

Edge computing [12] was born to push the computation and storage of data from the server to clients or edge devices, reducing the load on the server. Furthermore, the computation is moved to or near the data generation site, which saves huge amounts of communication costs and reduces computational latency. Not only that, the process of exchanging data between the server and the clients no longer takes place, greatly increasing the user's data privacy.

In the spirit of edge computing, *federated learning* (FL) [18] was born, to train a machine learning model on separate data sets, distributed on edge devices [21]. Specifically, the goal of the FL system is to find a global parameter w_G^* such that the system error is minimized:

$$\begin{aligned} w_G^* &= \arg\min_{w_G} f_{global}(w_G) \\ &= \arg\min_{w_G} \frac{1}{n}\sum_{i=1}^{n} f_{local}(w_i) \end{aligned} \tag{1}$$

where, n is the number of clients participating in the system, f_{global} is the error of the whole system, $f_{local}(w_i)$ is the error of the client i when it use the parameter w_i.

This training method not only inherits the benefits of edge computing (lowering hardware costs and latency, increased data security) but also has the potential to increase personalization for each user. However, the clients participating in the system are often heterogeneous in terms of storage and computing capacity. The data distributed across these clients is also not uniformly distributed (non-IID). This creates two main challenges of an FL system: the Systemic Challenge and the Statistical Challenge [15]. As for the statistical challenge, [22] research indicates that FL's traditional algorithm - Federated Averaging (FedAvg) suffers from severe performance loss on non-IID data. This study aims to partially solve the statistical problem. Specifically, the study focuses on improving performance as well as personalization in prediction on non-IID data.

In this study, we improve the performance of the FL system on non-IID data by meta-learning (ML) [9] and personalization layer (PL) technique. Bilevel optimization-based ML algorithms can generate a quick adaptive initialization on the new task. The use of ML algorithms in training aims to obtain a global parameter that can adapt quickly on the new data set of each user. Meanwhile, the use of PL techniques to maintain part of the deep neural network at the edge devices further improves personalization.

Contribution. The study proposes the FedMeta-Per algorithm, a combination of ML algorithms (MAML [7], Meta-SGD [16]) and PL techniques (FedPer [1], LG-FedAvg [17]), which helps to increase accuracy and personalize for each user. Experimentally, the research shows the effectiveness in increasing the accuracy as well as the personalization of the proposed algorithm compared with the algorithm FedAvg; algorithms that incorporate ML into FL [3] (FedMeta(MAML), FedMeta(Meta-SGD)) and algorithms that use PL (FedPer) on a FL system of 50 users, containing the data of two datasets MNIST [5] and CIFAR-10 [13] respectively.

2 Related Work

Since the FedAvg algorithm was first introduced in 2017, a lot of work has been done to improve the accuracy of the system on non-IID data. Here, the study reviews a few typical works in the realization of this goal.

Meta-learning based approach. ML is a new learning method that allows the learning model to increase experience by performing many different tasks in the same task distribution. This results in ML models being able to adapt quickly on new tasks after only a few training steps with limited training data. This is an important discovery, playing a role in bringing machine learning closer to human learning [8].

In the context of non-IID data, each user, with different needs, generates datasets with very different distributions. If we consider each user as a task, training the global model on a set of users (FL system) is equivalent to training the machine learning model on an task distribution (ML approach). Research [3] proposes framework FedMeta and experimentally illustrates by two algorithms FedMeta(MAML), FedMeta(Meta-SGD). The results show that the global model has a fast adaptability, faster convergence than FedAvg on the LEAF dataset [2]. The study [6] proposes the combination of MAML on the CIFAR-10 dataset and the study [11] proposes the combination Reptile [19] on the EMNIST-62 dataset [4] to the FL system to increase personalization on each client.

An FL system, on the other hand, will have new clients over time. One downside for systems that use ML in training is that they treat clients that have been around for a long time in the system and those that have just joined the system. Therefore, although global models are able to adapt quickly to new data, their performance can be further improved.

Personalization layer approach. To capture the data properties, which are highly personalized on different users, dividing the deep neural network into personalization layers and base layers, and maintaining the personalization layers at each client (base layers will be co-trained by the clients) is suggested. Accordingly, the study [1] proposes the FedPer algorithm with the idea of maintaining fully-connected layers of the deep neural network at the clients with the goal of increasing personalization. Experimentally, FedPer achieved much higher results than FedAvg when tested on the data sets CIFAR-10 and CIFAR-100. In addition, the study [17] with the algorithm LG-FedAvg proposes to maintain the feature extraction layers of the deep neural network at the clients with the goal of capturing the features of each different clients. The results obtained are even better than FedPer in most cases when testing on a system consisting of 10 to 100 users containing data of the data sets CIFAR-10, CIFAR- 100 and Omniglot [14].

The similarity between these two methods lies in the training base layers. These layers are trained and synthesized using the FedAvg algorithm. Therefore, they are not able to adapt quickly on the new data set. In addition, during inference phase, the FedPer algorithm takes the average of the parameters of the personalization layers, then combines with the parameters of the base layers to conduct the test. The parameter of personalization layers, which is highly personalized, is now averaged, which can cause the classification efficiency to be significantly reduced when the data is strong non-IID. The algorithm LG-FedAvg proposes to perform an ensemble test between personalization layers. However, only personalization layers trained on the same data as the test data will really perform well. Therefore, in some cases, ensemble test results are not high because there are too many personalization layers that have never worked with the same data as the data during the test.

3 Proposed Method

The proposed method of this study is a combination of maintaining personalization layers at the edge device of the algorithm FedPer and using meta-learning in training base layers of the model. With this combination, the research aims at two goals: (1) - Fast adaptability on new dataset of ML algorithms, (2) - High personalization ability of PL techniques.

Fast adaption. As mentioned above, the base layers of algorithms using PL techniques are not really strong because they are trained on the FedAvg algorithm, which leads to them being slow to adapt to the data set. new. The training of base layers by ML algorithms is expected to provide these base layers with the ability to quickly adapt on the data set of clients, overcoming the disadvantages of algorithms using PL techniques.

High personalization. The personalization of the FL system combined with ML comes from the fact that the system allows the global model to perform a fine-tune on part of the client data before going into testing on the rest of the client data. there. Allowing to maintain a portion of the deep neural network (personalization layers) on each client also greatly increases the ability to personalize each user. Here, FedMeta-Per proposes a combination of the two methods of personalization enhancement mentioned above, which is expected to further increase the personalization of the FL system.

Specifically, the deep neural network is divided into two parts, the base layers include feature extraction layers, and the personalization layers contain fully-connected layers. The base layers are trained against two ML algorithms (MAML and Meta-SGD). The personalization layers are maintained at separate clients. With this combination, we call our proposed algorithm FedMeta-Per.

Algorithm 1 FedMeta-Per (Server)

```
Initialize $w_B^0$ for MAML or $(w_B^0, \alpha_B^0)$ for Meta-SGD
for $t = 0, 1, 2, ...$ do
    Sample a subset $C_t$ of $m$ clients
    for $c_i \in C_t$ do
        $w_{B(i)}^{t+1} \leftarrow$ ModelTrainingMAML$(c_i, w_B^t)$ for MAML
        $(w_{B(i)}^{t+1}, \alpha_{B(i)}^{t+1}) \leftarrow$ ModelTrainingMetaSGD$(c_i, w_B^t, \alpha_B^t)$ for Meta-SGD

    $n_i = \left|\mathcal{D}_{train(i)}^{query}\right|, N_m = \sum_{i=0}^m n_i$
    Aggregate $w_B^{t+1} = \sum_{i=0}^m \frac{n_i}{N_m} w_{B(i)}^{t+1}$ for MAML
    Aggregate $(w_B^{t+1}, \alpha_B^{t+1}) = \sum_{i=0}^m \frac{n_i}{N_m} (w_i^{t+1}, \alpha_{B(i)}^{t+1})$ for Meta-SGD
```

Training phase. At the server, the algorithm 1 is implemented to coordinate the training activities. These operations include: Parameter initialization of base layers; Distribute this parameter to clients; Synthesize the parameters of the new global base layers. This process is summarized in Table 1. For the MAML algorithm, the base layers parameters include only part of the global model parameters (w_B^t at round t). Meanwhile, the parameter of the Meta-SGD algorithm's base layers contains an extra part of the learning rate at each client (α_B at round t), which is the same size as w_B^t. This learning rate value will be well explained in the training at the clients. These values will be distributed to a subset C_t of m clients to conduct distributed training. The clients then perform training on their own dataset and return the parameters of the base layers ($w_{B(i)}^{t+1}$ for MAML and $(w_{B(i)}^{t+1}, \alpha_{B(i)}^{t+1})$ for Meta-SGD at round t) for the server. The parameters of the new base layers are summarized as follows:

$$\text{MAML: } w_B^{t+1} = \sum_{i=0}^{m} \frac{n_i}{N_m} w_{B(i)}^{t+1} \tag{2}$$

$$\text{Meta-SGD: } (w_B^{t+1}, \alpha_B^{t+1}) = \sum_{i=0}^{m} \frac{n_i}{N_m} (w_i^{t+1}, \alpha_{B(i)}^{t+1}) \tag{3}$$

where, n_i is the number of data samples of the client's query set $c_i \in C_t$, N_m is the total number of data samples on the query set of m of the client participating in global training.

Table 1: Server activity summary at round t^{th}

	Algorithm	
	MAML	Meta-SGD
Send	w_B^t	(w_B^t, α_B^t)
Receive	Weights: $w_{B(i)}^{t+1}, i \in [1, m]$	Parameters: $(w_{B(i)}^{t+1}, \alpha_{B(i)}^{t+1}), i \in [1, m]$
Aggregate	w_B^{t+1}	$(w_B^{t+1}, \alpha_B^{t+1})$

At the clients, one of two algorithms 2 and 3 are implemented, corresponding to the training method of the type MAML and Meta-SGD . To train the model on bilevel optimization-based ML algorithms , the client data is divided into support set ($\mathcal{D}_{train(i)}^{support}$) and query set ($\mathcal{D}_{train(i)}^{query}$). The training steps at the client include: Merge parameters of base layers and personalization layers to form complete model parameters; Perform training according to ML algorithms; New model parameter resolution.

Specifically, for clients that train on the MAML algorithm, at the t global training step, client c_i will receive an initialization weight w_B^t of the base layers sent by the server. Next, c_i merges the weight of the base layers w_B^t with the weight of the personalization layers $w_{P(i)}^t$ to get the weight of the complete model w_i^t . w_i^t is then trained as follows:

$$\text{Train: } \hat{w}_i^{t+1} \leftarrow w_i^t - \alpha \nabla_{w_i^t} f_{local}\left(w_i^t, \mathcal{D}_{train(i)}^{support}\right) \tag{4}$$

Algorithm 2 FedMeta-Per (MAML Client)

```
**ModelTrainingMAML**($c_i, w_B^t$):
Sample support set $\mathcal{D}_{train(i)}^{support}$ and query set $\mathcal{D}_{train(i)}^{query}$
**if** $t = 0$ **then**
    Initialize $w_{P(i)}^t$
**else**
    Load $w_{P(i)}^t$ from the external memory
$w_i^t \leftarrow w_B^t \bigoplus w_{P(i)}^t$          ▷ Merge $w_B^t$ and $w_{P(i)}^t$ to form $w_i^t$
Training:
    $\hat{w}_i^{t+1} \leftarrow w_i^t - \alpha \nabla_{w_i^t} f_{local} \left( w_i^t, \mathcal{D}_{train(i)}^{support} \right)$
    $w_i^{t+1} \leftarrow w_i^t - \beta \nabla_{w_i^t} f_{local} \left( \hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query} \right)$
$w_{B(i)}^{t+1}, w_{P(i)}^{t+1} \leftarrow w_i^{t+1}$          ▷ Resolve $w_i^{t+1}$ to form $w_{B(i)}^{t+1}$ and $w_{P(i)}^{t+1}$
Send $w_{B(i)}^{t+1}$ to server and store $w_{P(i)}^{t+1}$
```

$$\text{Meta-train: } w_i^{t+1} \leftarrow w_i^t - \beta \nabla_{w_i^t} f_{local} \left(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query} \right) \tag{5}$$

where, $\alpha, \beta \in [0, 1]$ are the learning rates.

For clients that use the `Meta-SGD` algorithm in training, the base layers of the deep neural network include two parameters: the weight of the base layers w_B^t and the learning rate of the base layers α_B^t; personalization layers of the network include two parameters $w_{P(i)}^t$ and $\alpha_{P(i)}^t$ (same size as $w_{P(i))}^t$). Parameter merging also takes place between the parameters of base layers and personalization layers to form the complete parameter set w_i^t and α_i^t. It should be noted that `Meta-SGD` configures α as a learning rate array of the size of the model weight and treats α as a trainable parameter. Thus, the algorithm perform element-wise multiplication between α and the derivative of the error function in the Equation 6 and performs the derivative with respect to α in the Equation 7.

$$\text{Train: } \hat{w}_i^{t+1} \leftarrow w_i^t - \alpha_i^t \circ \nabla_{w_i^t} f_{local} \left(w_i^t, \mathcal{D}_{train(i)}^{support} \right) \tag{6}$$

$$\text{Meta-train: } (w_i^{t+1}, \alpha_i^{t+1}) \leftarrow (w_i^t, \alpha_i^t) - \beta \nabla_{(w_i^t, \alpha_i^t)} f_{local} \left(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query} \right) \tag{7}$$

Inference phase. Based on the testing process of the [17] study, we divide into two types of users: local clients and new clients. For local clients, private deep classes are reused to achieve a high degree of personalization. For new clients, we do not use ensemble testing techniques as suggested by the [17] study, but for each of the previously built personalization layers combined with base layers to operate on the new dataset. During the fine-tune process, the client selects the parameter set for the smallest error. Then use this set of parameters to process test data.

Algorithm 3 FedMeta-Per (Meta-SGD Client)

```
**ModelTrainingMetaSGD**($c_i, w_B^t, \alpha_B^t$)**:**
Sample support set $\mathcal{D}_{train(i)}^{support}$ and query set $\mathcal{D}_{train(i)}^{query}$
**if** $t = 0$ **then**
    Initialize $(w_{P(i)}^t, \alpha_{P(i)}^t)$
**else**
    Load $(w_{P(i)}^t, \alpha_{P(i)}^t)$ from the external memory
$w_i^t \leftarrow w_B^t \bigoplus w_{P(i)}^t$                ▷ Merge $w_B^t$ and $w_{P(i)}^t$ to form $w_i^t$
$\alpha_i^t \leftarrow \alpha_B^t \bigoplus \alpha_{P(i)}^t$                ▷ Merge $\alpha_B^t$ and $\alpha_{P(i)}^t$ to form $\alpha_i^t$
Training:
    $$\hat{w}_i^{t+1} \leftarrow w_i^t - \alpha_i^t \circ \nabla_{w_i^t} f_{local}\left(w_i^t, \mathcal{D}_{train(i)}^{support}\right)$$
    $$(w_i^{t+1}, \alpha_i^{t+1}) \leftarrow (w_i^t, \alpha_i^t) - \beta \nabla_{(w_i^t, \alpha_i^t)} f_{local}\left(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query}\right)$$
$w_{B(i)}^{t+1}, w_{P(i)}^{t+1} \leftarrow w_i^{t+1}$                ▷ Resolve $w_i^{t+1}$ to form $w_{B(i)}^{t+1}$ và $w_{P(i)}^{t+1}$
$\alpha_{B(i)}^{t+1}, \alpha_{P(i)}^{t+1} \leftarrow \alpha_i^{t+1}$                ▷ Resolve $\alpha_i^{t+1}$ to form $\alpha_{B(i)}^{t+1}$ và $\alpha_{P(i)}^{t+1}$
Send $(w_{B(i)}^{t+1}, \alpha_B^{t+1})$ to server and store $(w_{P(i)}^{t+1}, \alpha_{P(i)}^{t+1})$
```

4 Numerical Experiments

Dataset. The study evaluates the proposed algorithm on 50 clients containing two datasets MNIST [5] and CIFAR-10 [13], respectively. To simulate non-IID data, each client is configured to contain only 2/10 data subclasses, the amount of data between clients is uneven. At each client, the data is divided between the support and query sets at a ratio of 2:8. Data statistics are presented in Table 2.

Table 2: Statistics on MNIST and CIFAR-10 (non-IID data)

Dataset	#clients	#samples	#classes	#samples/client				#classes/client
				min	mean	std	max	
MNIST	50	69,909	10	135	1,398	1,424	5,201	2
CIFAR-10	50	52,497	10	506	1,049	250	1,986	2

Metrics. The study evaluates the model through the following metrics: Accuracy in correlation with all data points (acc_{micro}), Accuracy in correlation with all clients (acc_{macro}) and F1-score ($F1_{macro}$).

Experiments. This study performs centralized and distributed experiments on the same model architecture. For distributed training, the study experimented on algorithms `FedAvg`, `FedPer`, `FedMeta(MAML)`, `FedMeta(Meta-SGD)`. The results are compared with the proposed algorithm `FedMeta-Per`. For a fair comparison, the study allows the global model obtained by the FedAvg and FedPer algorithms to perform a fine-tune one or several times on the client's support set during the inference phase. This is the idea of FedAvgMeta and FedPerMeta algorithms.

Details of the experimental settings can be found in Appendix A.

5 Results & Discussion

Centralized & Decentralized. The results of centralized training and distributed training (FedAvg on IID and non-IID data) are presented in Table 3. It is easy to see that the centralized training model achieves higher results on all metrics. Results obtained on FedAvg (IID data) are 7% to 8% lower because the global model captures the features indirectly (through parametric averaging at the server). However, when the input data is non-IID, the metrics decrease from 9% to 13% on MNIST and do not reach convergence on CIFAR-10. Metrics on non-IID data also differ quite a lot (3% - 6% on MNIST and 3% - 4% on CIFAR-10) compared to differences on IID data (below 1%). This proves that the uneven data distribution across the clients has adversely affected the classification quality.

Table 3: Classification results (%) of centralized and decentralized (FedAvg on IID and non-IID data) using MNIST and CIFAR-10 datasets. Best results per metrics are boldfaced.

		acc_{micro}	acc_{macro}	$F1_{macro}$
MNIST	Centralized	**97.07**	-	**97.04**
	FedAvg (IID data)	90.36	90.34±2.24	90.12±2.29
	FedAvg (local client)	85.03	82.14±14.76	79.43±16.83
	FedAvg (new client)	83.92	81.69±19.71	77.66±22.54
CIFAR-10	Centralized	**61.91**	-	**61.72**
	FedAvg (IID data)	53.83	53.83±3.14	53±3.21
	FedAvg (local client)	19.02	19.29±25.11	16.85±23.92
	FedAvg (new client)	24.63	24.83±22.57	20.52±20.45

Convergence ability. The results on metrics of the FedMeta-Per algorithm and the algorithms used in the comparison are presented in Table 4. To solve the problem of non-IID data, the technique of fine-tune localizes the FL system in a simple form (algorithm FedAvgMeta), in order to improve the adaptability and personalization of the model on the new data set. However, the results obtained in Table 4 as well as in Figure 2 are not very satisfactory (low convergence on the MNIST dataset and not on the CIFAR-10 dataset). The reason for FedAvgMeta not to converge is that the adaptability of the global model to the new dataset is very low. The PL technique is also used in the form of two algorithms FedPer and FedPerMeta. However, the results obtained are even worse than FedAvg and

FedAvgMeta. This is understandable given that the modeling architecture used by the [1] study is a pre-trained network MobileNet-v1 [10] - a more complex deep neural network of this study a lot (see Appendix A.2). This also proves that using PL technique is not enough in handling non-IID data.

Table 4: Classification results (%) of FedAvg, FedPer and FedMeta-Per using MNIST and CIFAR-10 datasets. Best results per metrics are boldfaced.

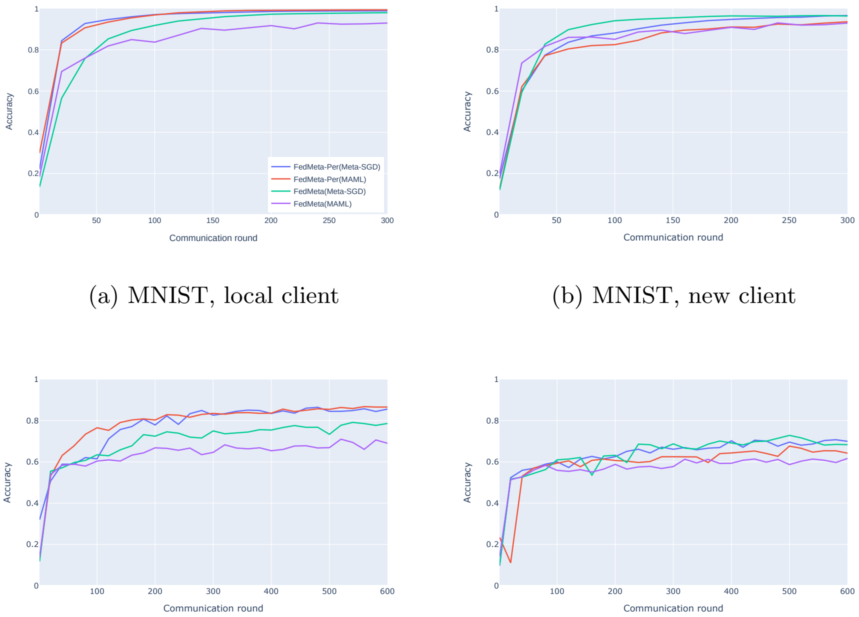
		CIFAR-10			MNIST		
		acc_{micro}	acc_{macro}	$F1_{macro}$	acc_{micro}	acc_{macro}	$F1_{macro}$
local client	FedAvg	19.02	19.29±25.11	16.85±23.92	85.03	82.14±14.76	79.43±16.83
	FedPer	13.22	12.99±19.39	10.52±14.91	77.29	75.48±14.84	72.32±15.99
	FedAvgMeta	40.3	38.47±31.52	33.81±30.61	84.84	81.56±16.68	78.31±19.8
	FedPerMeta	18.57	17.48±22.55	14.54±18.67	75.91	74.11±16.2	71.22±16.77
	FedMeta(MAML)	69.02	68.76±14.86	61.14±20	92.99	91.14±5.99	90.16±6.28
	FedMeta(Meta-SGD)	78.63	78.73±11.59	72.87±18.31	98.02	96.35±4.62	95.80±5.51
	FedMeta-Per(MAML)	**86.6**	**86.52±6.31**	**85.33±6.77**	**99.37**	**99.12±1.29**	**98.94±1.6**
	FedMeta-Per(Meta-SGD)	85.61	85.68±7.22	85.08±7.32	98.92	98.15±3.32	98.20±2.94
new client	FedAvg	24.63	24.83±22.57	20.52±20.45	83.92	81.69±19.71	77.66±22.54
	FedPer	14.4	14.52±20.15	10.66±13.79	78.3	76.19±18.79	72.72±19.3
	FedAvgMeta	43.39	43.54±18	35.14±17.22	84.34	82.37±17.42	78.78±19.31
	FedPerMeta	13.33	13.57±19.62	10.05±13.17	77.47	75.56±20.33	72.60±21.37
	FedMeta(MAML)	61.69	61.64±12.49	50.76±19.2	92.96	91.88±5.88	90.02±7.34
	FedMeta(Meta-SGD)	68.36	67.89±15.11	60.24±21.52	96.39	93.53±8.39	89.31±14.56
	FedMeta-Per(MAML)	64.22	63.70±12.29	53.68±19.06	93.6	93.57±5.58	91.83±6.43
	FedMeta-Per(Meta-SGD)	**69.97**	**69.13±14.63**	**62.42±20.94**	**96.62**	**95.88±3.58**	**94.85±4.61**

When comparing the convergence ability of FedMeta and FedMeta-Per, because both algorithms have the participation of ML and there are similarities in the convergence of the two algorithms (Fig. 1). It proves that ML has a certain impact on the FL system.

The ability of ML in the proposed algorithm in comparison with algorithms FedAvg, FedAvgMeta, FedPer and FedPerMeta is shown in Figure 2. Accordingly, ML provides the system with fast adaptability on new clients, leading to FedMeta-Per convergence much faster than the old algorithms, reaching convergence thresholds from about 60% to 85% on CIFAR-10 and about 20% improvement in accuracy on MNIST. More specifically, this convergence comes from performing a single fine-tune step on 20% of the data at the client, once again demonstrating the very fast adaptability of the FL-ML system.

In addition, thanks to the construction of good personalization layers, the testing process on local clients also achieves a higher degree of convergence when comparing the proposed algorithm with FedMeta (Figure 1). However, in front of a strange data distribution of new clients, these two algorithms have almost equivalent convergence ability.

Personalization ability. Observe the standard deviation of the algorithms FedMeta, FedAvg and FedPer in Table 4, we can see the standard deviation in the results of FedMeta is much smaller than the rest of the algorithms. It shows that the personalization of the FL system has been improved many times thanks to the training of ML algorithms as well as the fine-tune execution on the support



(a) MNIST, local client

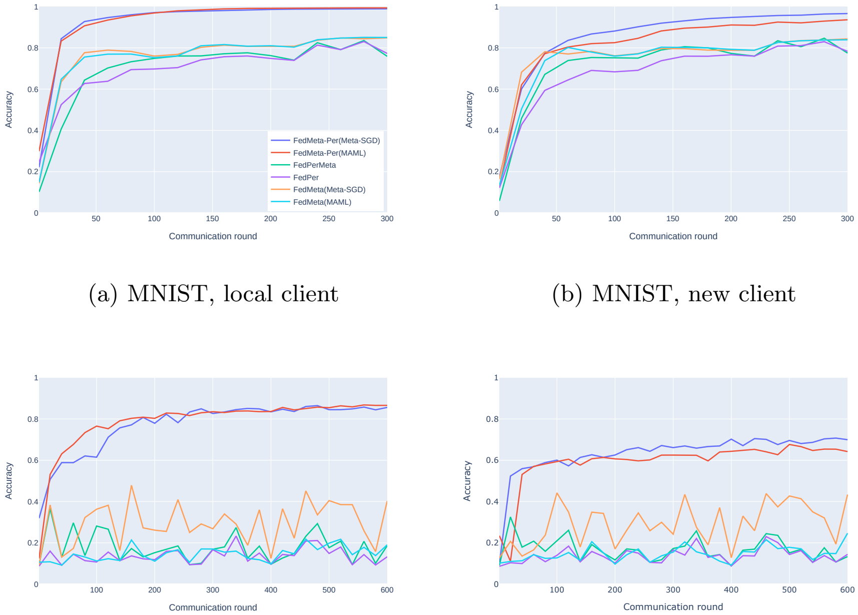
(b) MNIST, new client

(c) CIFAR-10, local client

(d) CIFAR-10, new client

Fig. 1: acc_{micro} of FedMeta-Per and FedMeta. The end result and convergence of FedMeta-Per is significantly higher than that of FedMeta on local clients. On new clients, even though FedMeta-Per reaches a higher value, the difference in convergence is not so great.

set to capture the characteristics of a new data set. However, as mentioned above, this ability can completely be improved. With new suggestions during inference phase, FedMeta-Per further reduces the standard deviations on the scales. Specifically, after combining the fine-tune of the model on the support set of local data and maintaining the personalization layer at each client, for the local clients, the standard deviation decreases from 4% to 14% on CIFAR-10 and from 1% to 5% on MNIST while the averages are still superior. On new clients, the mean values are higher but the standard deviation in some cases is not actually smaller than the previous algorithms. This is because the personalization layers in the deep neural network of new clients have not really matched that user yet. However, over time, when new clients participate in one or a few steps of global training, they will build up good personalization layers for themselves, resulting in the metrics reaching the same value as local clients.



(a) MNIST, local client

(b) MNIST, new client

(c) CIFAR-10, local client

(d) CIFAR-10, new client

Fig. 2: acc_{micro} of FedMeta-Per, FedAvg and FedPer. FedMeta-Per's convergence and accuracy are superior to those of FedAvg and FedPer algorithms.

6 Conclusion

Through the research process, we combine ML algorithms and PL techniques into the FL system, to improve the accuracy as well as the ability to personalize for each user of the system on non-IID data. Experimentally, we prove the effectiveness of the proposed algorithm FedMeta-Per compared with algorithms FedAvg, FedPer, FedMeta over 50 users with two datasets CIFAR-10 and MNIST. In which, it can be explained that achieving high results is based on two inherited factors: (1) - Fast adaptability on new data set of the proposed algorithm inherited from ML algorithms, (2) - High personalization ability for each user inherited from personalization layers of PL technique and model fine-tuning on support set of ML algorithms.

In the future, bilevel optimization-based algorithms (FO-MAML [7], iMAML [20], Reptile) can be integrated into the system. The search and clustering of users so that each user finds the best set of partial parameters should also be further studied to increase the performance of the system. In addition, the issue of information security between the server and the client should also be considered to make the system more practical.

References

1. Arivazhagan, M.G., Aggarwal, V., Singh, A.K., Choudhary, S.: Federated learning with personalization layers. arXiv preprint arXiv:1912.00818 (2019)
2. Caldas, S., Duddu, S.M.K., Wu, P., Li, T., Konečný, J., McMahan, H.B., Smith, V., Talwalkar, A.: Leaf: A benchmark for federated settings. arXiv preprint arXiv:1812.01097 (2018)
3. Chen, F., Luo, M., Dong, Z., Li, Z., He, X.: Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:1802.07876 (2018)
4. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1702.05373 (2017)
5. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
6. Fallah, A., Mokhtari, A., Ozdaglar, A.: Personalized federated learning: A meta-learning approach. arXiv preprint arXiv:2002.07948 (2020)
7. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. pp. 1126–1135. PMLR (2017)
8. Harlow, H.F.: The formation of learning sets. Psychological review **56**(1), 51 (1949)
9. Hospedales, T., Antoniou, A., Micaelli, P., Storkey, A.: Meta-learning in neural networks: A survey. arXiv preprint arXiv:2004.05439 (2020)
10. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
11. Jiang, Y., Konečný, J., Rush, K., Kannan, S.: Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:1909.12488 (2019)
12. Khan, W.Z., Ahmed, E., Hakak, S., Yaqoob, I., Ahmed, A.: Edge computing: A survey. Future Generation Computer Systems **97**, 219–235 (2019)
13. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
14. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science **350**(6266), 1332–1338 (2015)
15. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**(3), 50–60 (2020)
16. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835 (2017)
17. Liang, P.P., Liu, T., Ziyin, L., Allen, N.B., Auerbach, R.P., Brent, D., Salakhutdinov, R., Morency, L.P.: Think locally, act globally: Federated learning with local and global representations. arXiv preprint arXiv:2001.01523 (2020)
18. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
19. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
20. Rajeswaran, A., Finn, C., Kakade, S.M., Levine, S.: Meta-learning with implicit gradients. Advances in neural information processing systems **32** (2019)
21. Yin, X., Zhu, Y., Hu, J.: A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. ACM Computing Surveys (CSUR) **54**(6), 1–36 (2021)

22. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
23. Zhu, H., Xu, J., Liu, S., Jin, Y.: Federated learning on non-iid data: A survey. arXiv preprint arXiv:2106.06843 (2021)

A Experimental Details

A.1 Datasets

The notation $C_{train} = \{c_1^{train}, ..., c_{50}^{train}\}$ is the set of clients used in training, $C_{test} = \{c_1^{test}, ..., c_{50}^{test}\}$ is the set of clients used in the test, N is the total number of data points, we have the number of training data points and test data points in turn is:

$$N_{train} = \sum_{i=1}^{50} \left| c_i^{train} \right| = 0.75N$$

$$N_{test} = \sum_{i=1}^{50} \left| c_i^{test} \right| = 0.25N$$

In ML implementation, notation $c_i^{train} = \{\mathcal{D}_{train(i)}^{support}, \mathcal{D}_{train(i)}^{query}\}$, $c_i^{test} = \{\mathcal{D}_{test(i)}^{support}, \mathcal{D}_{test(i)}^{query}\}$. We have the amount of data contained in the support and query sets of client i respectively:

$$N_{train/test(i)}^{support} = \left| \mathcal{D}_{train/test(i)}^{support} \right| = 0.2 \left| c_i^{train/test} \right|$$

$$N_{train/test(i)}^{query} = \left| \mathcal{D}_{train/test(i)}^{query} \right| = 0.8 \left| c_i^{train/test} \right|$$

Client $c_j^{test} \in C_{test}$ is called a local client if there exists a client $c_i^{train} \in C_{train}$ such that $p\left((x,y) \in c_j^{test}\right) = p\left((x,y) \in c_i^{train}\right)$. Otherwise, c_j^{test} is a new client if $p\left((x,y) \in c_j^{test}\right) \neq p\left((x,y) \in c_i^{train}\right)$ for all $c_i^{train} \in C_{train}$.

A.2 Model Architecture

The study uses two simple models for feature extraction and data classification for the CIFAR-10 and MNIST datasets.

CIFAR-10. The model takes input images of size $(32 \times 32 \times 3)$. Two convolution layers (kernel of size (5×5), channel numbers of 6 and 16 respectively) are used for feature extraction. Following each convolution layer is a `MaxPooing` layer of size (2×2). The classifier consists of three linear layers whose outputs are 120, 84 and 10 respectively. The activation functions used are `ReLU` and `Softmax`.

MNIST. The model receives flattened input images of size (1×784). Use two linear layers with outputs of 100 and 10 respectively. The activation functions used are `ReLU` and `Softmax`.

A.3 Hyper-parameters Searching

The hyper-parameters of the FL system of the study include: number of clients participating in training in a global training step ($\#clients/round$), number of local training steps ($\#epochs$), number of global training steps ($\#rounds$), amount of data in a data batch ($batch\_size$), number of personalization layers for algorithms using PL techniques ($\#per\_layers$) and the learning rate used in the optimization of deep neural networks by mini batch gradient descent.

To accommodate the limited configuration client hardware in the Horizontal FL scenario, this study limits $\#epochs = 1$ and $batch\_size = 32$. From the survey of FL experiments of recent studies, the number of clients participating in global training was selected as 2, 5 and 10, respectively. In which, $\#clients/round = 5$ gives a higher result and consumes an acceptable computational cost.

For installed deep learning networks, it is natural to maintain a base layer and a personalization layer for the neural network of the MNIST dataset. For the deep learning network used for the set CIFAR-10, $\#per\_layers \in \{1, 2, 3\}$ (calculated from the last linear layer). Experimental results show that using the last linear layer as the personalization layer and the remaining deep layers as the base layers gives the best results.

Except for the learning rate of each algorithm, the above hyper-parameters are kept fixed during training. Table 5 summarizes these hyper-parameters values.

Table 5: Fixed hyper-parameters in FL system

	#clients/round	#epochs	#rounds	batch_size	#per_layers
MNIST	5	1	300	32	1
CIFAR-10			600		

The learning hyper-parameters are searched in the range $(10^{-5}, 0.01)$ for each algorithm. Search results are presented in Table 6. Empty cells indicate that hyper-parameters cannot be found for the model to converge.

Table 6: Tuning learning rate for algorithms

	CIFAR-10	MNIST
FedAvg, FedAvgMeta	-	10^{-5}
FedPer, FedPerMeta	-	10^{-5}
FedMeta(MAML) (α, β)	$(0.01, 0.001)$	$(0.001, 0.001)$
FedMeta(Meta-SGD)(α, β)	$(0.001, 0.001)$	$(0.001, 5 \times 10^{-4})$
FedMeta-Per(MAML)(α, β)	$(0.001, 0.005)$	$(0.001, 0.001)$
FedMeta-Per(Meta-SGD)(α, β)	$(0.01, 0.01)$	$(0.001, 5 \times 10^{-4})$