Personalized Federated Learning with Contrastive Momentum

Sen Fu, Zhengjie Yang, *Student Member, IEEE,* Chuang Hu, *Member, IEEE,* Wei Bao, *Member, IEEE,*

Abstract—In this paper, we propose pFedMo, a personalized federated learning algorithm with contrastive momentum. In pFedMo, we design a score function to personalize worker models by distilling knowledge from the aggregator's representation model so as to address the non-i.i.d. issue. To accelerate the convergence, we leverage the momentum acceleration on both the worker side and the aggregator side. However, the typical momentum without personalization does not suit well for the worker models with personalization, influencing convergence performance. To address this, we develop a personalized/contrastive momentum method for efficient momentum acceleration. We provide mathematical proof for the convergence of pFedMo on non-i.i.d. data. Extensive experiments based on real-world datasets and IoT system are conducted, verifying that pFedMo outperforms existing mainstream benchmarks, and achieves up to 35.90% accuracy increase and 3.64x training time speedup under a wide range of settings.

Index Terms—Federated learning, Edge computing, Momentum, Contrastive Learning

1 Introduction

THE proliferation of smart devices, Mobile Edge Computing (MEC), and Artificial Intelligence (AI) technologies have stimulated the development of Internet of Things (IoT) and Industry 4.0. It advances all aspects of our modern life, such as smart home [1], smart health [2], and smart transportation [3], etc. With the abundance of data generated in IoT devices, transmitting huge data to a centralized server for Machine Learning (ML) activity costs huge communication burdens. Moreover, according to General Data Protection Regulation (GDPR) [4], individual users are sometimes not willing to share their sensitive data. To address above issues, Federated Learning (FL) emerges [5]. It allows workers to collaboratively train a generalized global model by transmitting only the model parameters while keeping the data locally. In the context of edge computing [6], as depicted in Fig. 1, it emerges as an ideal platform for the implementation of FL. In the FL framework, an edge server aggregates worker/local models from edge devices such as laptops, smartphones, and tablets, etc. to obtain an updated global model. This updated global model is then sent back to the edge devices for the next round of local training. However, due to non-i.i.d. data distribution among workers, the convergence performance of FL algorithms still degrades fast on highly heterogeneous data.

Recently, Personalized Federated Learning (PFL) [7] has demonstrated its advantage in addressing the fundamental challenges of FL on heterogeneous (non-i.i.d.) data. PFL analyzes the characteristics of models between different parties (e.g., worker to worker, worker to aggregator) or timelines (e.g., current model to previous model) to adapt the global model to better align with individual worker's characteristics. However, since each individual worker model is personalized solely on its own data distribution, it will introduce the workers' bias into the global aggregation phase, resulting in the global model less robust and inefficient [7]. To address this issue, we utilize Contrastive Learning (CL) [8],

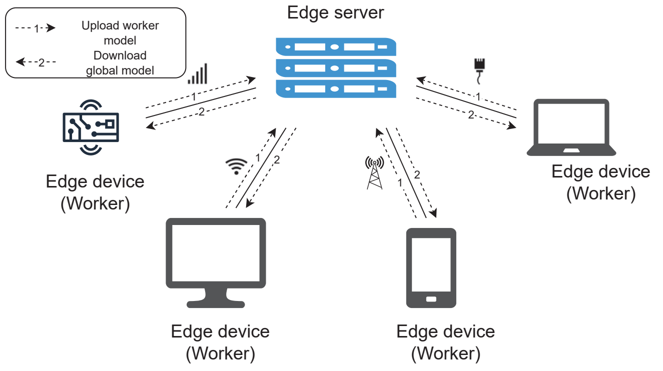


Fig. 1. Federated Learning in Mobile Edge Computing (MEC).

[9] to develop a new personalized FL framework. CL is a technique in machine learning, which encourages similar instances with short distance (e.g., reflected by small loss) while pushing away dissimilar instances with long distance. Apart from the training on workers, a representation model is trained at the aggregator using a publicly available i.i.d. dataset accessible only to the aggregator. A score function is designed to contrast the response-based knowledge (i.e., model prediction output) between worker models and the representation model. Since the representation model is trained on i.i.d. data, containing unbiased knowledge, we assign a higher weight to the representation model for worker with higher score, allowing it to contribute more to the training process. In this way, the unbiased knowledge is gradually distilled from the representation model, so the bias between each personalized worker model is countered.

However, the optimizer used in typical PFL algorithms is still inefficient, because these algorithms usually implement gradient descent or stochastic gradient descent, requiring many iterations to converge. Momentum is proved to be a valuable component for training models. Apart from the conventional gradient descent step, the momentum method conducts an additional momentum step [10] by adding a fraction γ of the difference between past model vector and current model vector of the objective function to accelerate the convergence. Nevertheless, utilizing a single aggregated momentum still causes problem. Since the model is personalized for each worker, such momentum does not suit all personalized worker models well, leading to inefficient momentum acceleration. To solve this problem, we develop personalized (contrastive) momentum which is in pair with the personalized model within each worker for better momentum acceleration. Based on the above integrated motivation, in this paper, we propose pFedMo, a personalized FL algorithm with contrastive momentum. It significantly enhances the performance under non-i.i.d. data distribution while preserving the efficient momentum acceleration.

Theoretically, we prove that pFedMo is convergent and has an $\mathcal{O}\left(\frac{1}{T}\right)$ convergence rate for smooth non-convex problems for a given T iterations. By utilizing CL, we design a new score function to characterize the contrast between the worker models and the representation model, which is then used in the personalization of momentum and model, and the aggregation phrase.

In the experiment, we compare the performance of pFedMo with three categories of FL algorithms: ① single-momentum FL (FedMom [11], SlowMo [12], FedNAG [13], and Mime [14]), ② double-momentum FL (FastSlowMo [15], DOMO [16], and FedADC [17]), and ③ no-momentum FL (FedProx [18],FedAvg [5]). The experiment is implemented on five real-world datasets (MNIST [19], CIFAR-10 [20], ImageNet [21], UCI-HAI [22], and Fire-detection-Dataset [23]) with six machine learning models (linear regression, logistic regression, LeNet5 [24], VGG16 [25], ResNet18 [21], and FireNet [26]). We also conduct experiments on the hyper-parameters that could potentially impact the performance of pFedMo, including the aggregation period τ, momentum factor γ, personalization temperature π, and the number of workers N. The outcomes of these experiments align with our initial theoretical analyses. Furthermore, we develop and deploy a real-world FL system. The system aims to assess the overall training duration in real-world scenarios, encompassing communication delays, computing delays both at the worker and server ends, as well as other associated overhead delays. The contributions of this paper are summarized as follows.

- We have proved that pFedMo is convergent and has an $\mathcal{O}\left(\frac{1}{T}\right)$ convergence rate for smooth non-convex problems for a given T iterations under non-i.i.d. data.
- The experimental results illustrate that pFedMo increases the training accuracy by 0.21-35.90% compared to nine state-of-the-art (SOTA) benchmark FL algorithms under a wide range of settings.
- The experiments implemented in the real-world FL system further illustrate that pFedMo increases training speed by 1.09-3.64x compared to benchmarks under a wide range of settings.

The rest of the paper is organized as follows. We provide the related worker in Section 2. The pFedMo design is described in Section 3. For Section 4, it provides the theoretical results including the convergence analysis of pFedMo. Section 5 provides our experiment result which shows the convergence performance of pFedMo over other mainstream momentum-based algorithms. The conclusion are written in Section 6.

2 Related Work

2.1 Personalized Federated Learning

Personalized Federated Learning (PFL) [7] is a general concept that facilitates the personalization for individual workers in FL environment. One of the main strategies for PFL is global model personalization [7]. It has demonstrated its advantage in addressing the issue of poor convergence on heterogeneous data.

In global model personalization, algorithms first train a single global FL model. This trained global model is then personalized for each worker through a local adaptation step. Such approach aims to train a stronger global model so as to improve the subsequent personalization performance on workers. FedProx [18] and FedProxVR [27] introduce a proximal term in the local loss function which helps assess the dissimilarity between global and local models for local model personalization. SCAFFOLD [28] measures the weight divergence between global and local models and incorporates a variance reduction term in the local loss function to correct worker updates. FedHealth [29] adopts a two-step algorithm which first trains a global model and then transfers back to each worker for personalized worker model adaption with its own dataset using Transfer Learning (TL) [30].

In contrast to global model personalization, another approach is to directly build personalized models during the FL model aggregation process. This can be achieved through various techniques. With the help of Knowledge Distillation (KD) [31], FedMD [32] starts with a pre-trained model on a public dataset, and then transfers and distills knowledge from this model to enable workers to design personalized models on their own datasets. Different from the traditional FL which learns models for the same task, in MOCHA [33], authors introduce a federated Multi-task Learning (MTL) framework where each worker learns a personalized model tailored to its specific task. Federated MTL [34] aims to learn distinct tasks across different workers by analyzing the relationships between models. Therefore, the model itself is personalized.

In summary, PFL is an effective approach for personalizing the worker models while maintaining the generalization performance of global model. Nevertheless, the optimizer used in these algorithms (gradient descent/stochastic gradient descent) still leads to slow convergence, requiring many iterations to converge.

2.2 Momentum in Federated Learning

Momentum [35] is a method that enhances the gradient descent by incorporating a weight γ of the difference between past and current model vectors. In the classical centralized machine learning scenario, the Polyak's momentum update rule is defined as follows:

$$\boldsymbol{m}(t) = \gamma \boldsymbol{m}(t-1) - \eta \nabla F(\boldsymbol{w}(t-1)), \tag{1}$$
$$\boldsymbol{w}(t) = \boldsymbol{w}(t-1) + \boldsymbol{m}(t), \tag{2}$$

where γ is the momentum factor with the range $\gamma \in [0,1)$, t represents the iteration of the update, $\boldsymbol{m}(t)$ denotes the momentum term at iteration t with the initial value $\boldsymbol{m}(0) = \mathbf{0}$, and $\boldsymbol{w}(t)$ is the model parameter at iteration t. Momentum selectively amplifies updates for dimensions with consistent gradient directions and dampens updates for dimensions with changing gradient directions. Consequently, momentum accelerates convergence and diminishes oscillation [10], [36]. In [10], [37], a variation of momentum known as Nesterov Accelerated Gradient (NAG) is introduced. NAG enhances the gradient calculation by approximating the gradient based on the next estimated parameter position, which is given by $\nabla F(\boldsymbol{w}(t-1) + \gamma \boldsymbol{m}(t-1))$, rather than $\nabla F(\boldsymbol{w}(t-1))$ used in Polyak's momentum, i.e.,

$$\boldsymbol{m}(t) = \gamma \boldsymbol{m}(t-1) - \eta \nabla F(\boldsymbol{w}(t-1) + \gamma \boldsymbol{m}(t-1)), \tag{3}$$
$$\boldsymbol{w}(t) = \boldsymbol{w}(t-1) + \boldsymbol{m}(t). \tag{4}$$

This modification results in improved convergence performance. It is important to note that there are two commonly used equivalent representations of NAG. Let $\hat{\boldsymbol{w}}(t-1) = \boldsymbol{w}(t-1) + \gamma \boldsymbol{m}(t-1)$ and employ mathematical transformation, we can directly derive the second representation of NAG [38], [39] from (3) and (4), which will be used in this paper. Extensive research has been conducted on momentum in FL. Based on the incorporation of momentum, it can be classified into single-momentum algorithms [11], [12], [13], [14] and double-momentum algorithms [15], [16], [17]. Single-momentum algorithms utilize momentum acceleration either on the worker side or the aggregator side. On the other hand, double-momentum algorithms combine the momenta of both the worker and the aggregator, resulting in superior convergence performance compared to single-momentum algorithms. All of these algorithms strive to train a global model that achieves enhanced performance through the utilization of momentum acceleration.

In this paper, we aim to leverage the benefits of both PFL and momentum by combining these two methods synergistically.

3 Algorithm Design

3.1 Problem Formulation

We consider a conventional FL system where N workers and one aggregator are involved in the FL training process. All workers are expected to participate in the model training, which is within the scope of cross-silo FL [40]. Each worker, denoted by i, has its own local dataset with the number of data samples denoted by D_i. The total number of dataset for all workers can be then represented by $D = \sum_{i=1}^{N} D_i$. The target of FL is to find the stationary point x^* that minimizes the global loss function

$$\min_{x \in \mathbb{R}^d} F(\boldsymbol{x}) \triangleq \sum_{i=1}^{N} \frac{D_i}{D} F_i(\boldsymbol{x}) \tag{5}$$

TABLE 1
Key Notations

η	learning rate
τ	aggregation period
γ	momentum factor
s_i^k	worker i's score at k's aggregation
π	temperature of score
T	number of total local (worker) iterations indexed by t
K	number of total aggregations indexed by k
N	number of workers indexed by i
$\boldsymbol{y}_i^t$	worker i's momentum at iteration t
$\boldsymbol{x}_i^t$	worker i's model at iteration t
$\boldsymbol{y}_r^t$	representation momentum at iteration t
$\boldsymbol{x}_r^t$	representation model at iteration t
$\boldsymbol{y}_{i+}^t$	worker i's personalized momentum at iteration t
$\boldsymbol{x}_{i+}^t$	worker i's personalized model at iteration t
$\boldsymbol{x}_+^t$	global model at iteration t

where d is the dimension of $\boldsymbol{x}$; $F_i(\boldsymbol{x})$ is the local loss function at worker i and $F(\boldsymbol{x})$ is the global loss function at the aggregator. Since workers are distributed in geometric locations among the network, the data generated in each worker is typically influenced by its unique characteristics, local environments, and user preferences. Consequently, the dataset on each worker is heterogeneous and non-independent and identically distributed (non-i.i.d.) [41]. The key notations are summarized in Table 1.

3.2 pFedMo Algorithm

In this section, we propose a personalized FL with contrastive momentum algorithm, named as pFedMo, to solve formula (5). In pFedMo, the momentum acceleration is leveraged at each worker in every local iteration. In the aggregator, a score function is developed to calculate scores for all workers which are then used in the worker momentum and model personalization. The pFedMo algorithm is conduced in T local iterations with K aggregations, where $T = K\tau$. Here, τ represents the aggregation period, indicating that a global aggregation takes place after every τ local iterations.

3.2.1 Worker Update

At each local iteration t, each worker computes worker momentum update

$$\boldsymbol{y}_i^t \leftarrow \boldsymbol{x}_i^{t-1} - \eta \nabla F_i(\boldsymbol{x}_i^{t-1}) \tag{6}$$

and worker model update

$$\boldsymbol{x}_i^t \leftarrow \boldsymbol{y}_i^t + \gamma(\boldsymbol{y}_i^t - \boldsymbol{y}_i^{t-1}). \tag{7}$$

Equations (6) and (7) follow the Nesterov Accelerated Gradient (NAG) [37] which has demonstrated its faster convergence over Polyak's momentum [35]. The utilization of worker momentum accelerates the worker model training in every local training iteration.

3.2.2 Score Function

When $t = k\tau, k = 1, 2, \ldots, K$, the aggregator calculates scores for all workers, denoted as $\mathcal{S}_k = \{s_1^k, s_2^k, \ldots, s_N^k\}$, where $s_i^k \in [0,1]$ denotes the worker i's score at k's aggregation. The aggregator first trains a representation model based on a public i.i.d. dataset that can only be accessed

Algorithm 1 pFedMo algorithm

```
Input: $\tau, T = K\tau, \eta, \gamma$
Output: Final model parameter $\boldsymbol{x}_{+}^{T}$
 1: For each worker, initialize: $\boldsymbol{x}_i^0$ as same value for all $i$,
    and $\boldsymbol{y}_i^0 = \boldsymbol{x}_i^0$
 2: For the aggregator, initialize: $\boldsymbol{x}_r^0 = \boldsymbol{x}_i^0$, and $\boldsymbol{y}_r^0 = \boldsymbol{x}_r^0$
 3: for $t = 1, 2, \ldots, T$ do
 4:   For each worker $i = 1, \ldots, N$ in parallel:
 5:   Compute worker momentum $\boldsymbol{y}_i^t$ as (6)
 6:   Compute worker model $\boldsymbol{x}_i^t$ as (7)
 7:   if $t == k\tau$ where $k = 1, \ldots, K$ then
 8:     For each worker $i = 1, \ldots, N$ in parallel:
 9:     Send $\boldsymbol{y}_i^{k\tau}$ and $\boldsymbol{x}_i^{k\tau}$ to the aggregator
10:     For the aggregator:
11:     Compute scores $\mathcal{S}_k$ as (8)–(10) and in Algorithm 2
12:     Personalize each worker $i$'s momentum $\boldsymbol{y}_{i+}^{k\tau}$ as (11)
13:     Personalize each worker $i$'s model $\boldsymbol{x}_{i+}^{k\tau}$ as (12)
14:     Aggregate $\boldsymbol{x}_{+}^{k\tau} \leftarrow \sum_{i=1}^{N} \frac{D_i}{D} \boldsymbol{x}_{i+}^{k\tau}$
        //Aggregate generalized global model
15:     Set $\boldsymbol{y}_i^{k\tau} \leftarrow \boldsymbol{y}_{i+}^{k\tau}$ for each worker $i$
        //Distribute personalized worker momentum to
        workers
16:     Set $\boldsymbol{x}_i^{k\tau} \leftarrow \boldsymbol{x}_{i+}^{k\tau}$ for each worker $i$
        //Distribute personalized worker model to
        workers
17:   end if
18: end for
```

by the aggregator. Since worker models are trained on non-i.i.d. data while the representation model is trained on i.i.d. data (holding unbiased knowledge), we treat the output of the representation model as the ground truth. For fair comparison, the representation model is trained following the same update rules (replace subscript i with r in (6) and (7)) with the same number of iterations (i.e., τ iterations) as in workers.

We consider a dataset with C classes. At k's aggregation, where $k = 1, 2, \ldots, K$, the worker i's model and the representation model make predictions on a batch of B samples extracted from the testing dataset. The loss is calculated as

$$l_b = -\sum_{c=1}^{C} w_{p_{b,c}} \log \frac{\exp(q_{b,c})}{\sum_{j=1}^{C} \exp(q_{b,j})}, \tag{8}$$

where q is the predicted result obtained by worker model, p is the predicted result obtained by representation model, and w_p is the weight which is normalized from p by `softmax` [42], i.e.,

$$w_{p_{b,c}} = \texttt{softmax}(p_{b,c}) = \frac{\exp(p_{b,c})}{\sum_{j=1}^{C} \exp(p_{b,j})}. \tag{9}$$

Please note that in the Deep Neural Networks (DNN), the predicted result (p or q) is the vector containing the unnormalized logits for each class (with dimension C) obtained from the last fully connected layer before the normalization function (LogSoftmax or Sigmoid, etc.). Finally, we average the values of all $l_b, \forall b \in B$ to obtain the final loss of

Algorithm 2 Calculation of score algorithm

```
Input: $K, \mathcal{L}_k = \{\ell_1^k, \ell_2^k, \ldots, \ell_N^k\}$
Output: $\mathcal{S}_k = \{s_1^k, s_2^k, \ldots, s_N^k\}$
1: Initialize: $u_i = \ell_i^1$ for all workers
2: for $k = 1, 2, \ldots, K$ do
3:   For each worker $i = 1, \ldots, N$ in parallel:
4:   if $u_i < \ell_i^k$ then
5:     $u_i \leftarrow \ell_i^k$
6:   end if
7:   $s_i^k \leftarrow 1 - \frac{\ell_i^k}{u_i}$
8: end for
```

worker i in k's aggregation,

$$\ell_i^k = \frac{1}{B} \sum_{b=1}^{B} l_b \tag{10}$$

The calculation of ℓ_i^k comes from the spirit of Contrastive Learning (CL) [8], [9], where a small loss indicates that the worker prediction result is close to the most likely result while away from the least likely result (predicted by the representation model). We assign a high score to the worker with a small loss and vice versa. Please note that different from the typical cross-entropy, where the weight w is given by one-hot encoded information, i.e., the weight is binary (the only one true label is 1, and other false labels are all 0), in pFedMo, the weight is calculated in each global aggregation and is continuous, ranging from 0 to 1. By doing so, the score function measures the disparity between worker models and the representation mode, while the cross-entropy can only quantify the deviation between the model's predicted result and the ground truth label. In order to normalize the value of ℓ_i^k to the score s_i^k in the range of $[0, 1]$, we develop Algorithm 2 to calculate scores for all workers when global aggregation occurs. Please note that $\mathcal{S}_k$ plays a vital role in the personalization of worker momentum and model.

3.2.3 *Aggregator Update*

When $t = k\tau, k = 1, 2, \ldots, K$, the aggregator performs aggregator update, including ① computing scores $\mathcal{S}_k$ for all workers, ② personalizing each worker i's momentum $\boldsymbol{y}_{i+}^{k\tau}$, ③ personalizing each worker i's model $\boldsymbol{x}_{i+}^{k\tau}$, and ④ aggregating global model and re-distributing personalized momenta and models to workers. ① has been introduced in Section 3.2.2. Then, we use the scores $\mathcal{S}_k$ calculated by ① to operate ② and ③.

In ②, the worker i's personalized momentum $\boldsymbol{y}_{i+}^{k\tau}$ is updated as

$$\boldsymbol{y}_{i+}^{k\tau} \leftarrow (1 - \pi s_i^k) \sum_{i=1}^{N} \frac{D_i}{D} \boldsymbol{y}_i^{k\tau} + \pi s_i^k \boldsymbol{y}_r^{k\tau}, \tag{11}$$

where $\boldsymbol{y}_r^{k\tau}$ is the representation momentum at k's aggregation, and $\pi \in [0, 1]$ is the temperature hyper-parameter which determines the sensitivity of impact of s_i^k. The worker with a higher score implies that its local model update direction is more aligned with the update direction of the representation model. Therefore, we assign a high weight (large score) of representation momentum to this worker, allowing it to contribute more to the training process.

In ③, the worker i's personalized model $\boldsymbol{x}_{i+}^{k\tau}$ is update as

$$\boldsymbol{x}_{i+}^{k\tau} \leftarrow (1-\pi s_i^k)\sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{x}_i^{k\tau} + \pi s_i^k \boldsymbol{x}_r^{k\tau} + \sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{y}_i^{k\tau} - \boldsymbol{y}_{i+}^{k\tau}, \tag{12}$$

where $\boldsymbol{x}_r^{k\tau}$ is the representation model at k's aggregation. Please note that $\sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{x}_i^{k\tau}$ is the aggregated model without personalization. In conventional FL [5], this aggregated model is used as the global model and distributed back to all workers. However, our approach goes a step further by personalizing each worker's model based on it. In (12), we assign a higher weight to the representation model and a lower weight to the original aggregated model for workers with high scores. Since the representation model contains unbiased knowledge (trained on i.i.d. data), the bias between personalized worker models is countered. Additionally, We incorporate a contrastive/personalized momentum term $\sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{y}_i^{k\tau} - \boldsymbol{y}_{i+}^{k\tau}$ to further accelerate the personalized model update.

Please note it is necessary to personalize not only worker models but also worker momenta (operations ② and ③). This is because although personalized worker models help alleviate the non-i.i.d. problem by distilling knowledge from the representation model, it is also important to personalize the momentum for each worker to better suit each personalized worker model for better momentum acceleration. In this way, pFedMo significantly enhances the performance under non-i.i.d. data distribution while preserving the efficient momentum acceleration.

In ④, the aggregator aggregates the global model $\boldsymbol{x}_{+}^{k\tau}$ (Line 14). Then, pFedMo re-distributes the personalized worker momentum (Line 15) and model (Line 16) to all workers for the next round of training iteration.

4 Theoretical Results

4.1 Preliminaries

We assume $F_i(\cdot)$ satisfies the following standard conditions that are necessary in theoretical analysis [13], [27], [43].

Assumption 1. *(ρ-Lipschitz). The local loss function is ρ-Lipschitz.*

$$\|F_i(\boldsymbol{x}_1) - F_i(\boldsymbol{x}_2)\| \le \rho\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \forall \boldsymbol{x}_1, \boldsymbol{x}_2, i.$$

Assumption 2. *(β-smooth). The local gradient is β-smooth.*

$$\|\nabla F_i(\boldsymbol{x}_1) - \nabla F_i(\boldsymbol{x}_2)\| \le \beta\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \forall \boldsymbol{x}_1, \boldsymbol{x}_2, i.$$

Assumption 3. *(Bounded diversity). The variance of local gradient to global gradient is bounded.*

$$\|\nabla F_i(\boldsymbol{x}) - \nabla F(\boldsymbol{x})\| \le \delta_i, \forall \boldsymbol{x}, i.$$

By applying Triangle Inequality for $F_i(\cdot)$ in Assumptions 1 and 2, we obtain that global loss function $F(\cdot)$ also satisfies ρ-Lipschitz and β-smooth. For Assumption 3, we define $\delta \triangleq \sum_{i=1}^{N}\delta_i$.

4.2 Virtual Update

In Algorithm 1, in order to index the global aggregation, we introduce a concept "Interval", denoted by $[k]$, to divide the total T local iterations into K intervals ($T = K\tau$), i.e., interval $[k]$ contains τ local iterations ($t \in [(k-1)\tau, k\tau]$) and one global aggregation (k's aggregation). Inspired by [43], we introduce the concept "Virtual Update" as if the momentum and model are updated on the total training dataset D (equivalent to virtually centralized update). The rationale behind the virtual update is that the diversity of gradients among workers makes it challenging to directly bound the real update. Nevertheless, we can first bound the gap between real update and the virtual update, and then analyze the convergence of the virtual update so as to obtain the convergence of the real update. This becomes the roadmap of the convergence analysis which will be discussed in Section 4.3.

At the beginning of the interval $[k]$ when $t = (k-1)\tau$, we set the initial values for virtual update

$$\boldsymbol{y}_{[k]}^{(k-1)\tau} \leftarrow \boldsymbol{y}_{+}^{(k-1)\tau}, \tag{13}$$

$$\boldsymbol{x}_{[k]}^{(k-1)\tau} \leftarrow \boldsymbol{x}_{+}^{(k-1)\tau}. \tag{14}$$

Then, it will be conducted for τ time when $t \in ((k-1)\tau, k\tau]$ as

$$\boldsymbol{y}_{[k]}^{t} \leftarrow \boldsymbol{x}_{[k]}^{t-1} - \eta\nabla F(\boldsymbol{x}_{[k]}^{t-1}), \tag{15}$$

$$\boldsymbol{x}_{[k]}^{t} \leftarrow \boldsymbol{y}_{[k]}^{t} + \gamma(\boldsymbol{y}_{[k]}^{t} - \boldsymbol{y}_{[k]}^{t-1}). \tag{16}$$

We also define aggregated values $y^t = \sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{y}_i^t, \boldsymbol{x}^t = \sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{x}_i^t, \forall t$, and $\boldsymbol{y}_{+}^t = \sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{y}_{i+}^t, \boldsymbol{x}_{+}^t = \sum_{i=1}^{N}\frac{D_i}{D}\boldsymbol{x}_{i+}^t$, $\forall t = k\tau, k = 1, 2, \ldots, K$ that will be used in convergence analysis for convenient presentation.

4.3 Convergence Analysis

Following the rationale of virtual update, we first bound the gap between $\boldsymbol{x}^t$ and virtual update $\boldsymbol{x}_{[k]}^t$ in Theorem 1.

Theorem 1. *For any interval $[k]$, $\forall t \in ((k-1)\tau, k\tau]$, we have*

$$\|\boldsymbol{x}^t - \boldsymbol{x}_{[k]}^t\| \le f(t-(k-1)\tau), \tag{17}$$

where $f(z)$ is

$$f(z) = \eta\delta\left(C_3(\gamma C_1)^z + C_4(\gamma C_2)^z - \frac{1}{\eta\beta} - \frac{\gamma^2(\gamma^z - 1) - (\gamma-1)z}{(\gamma-1)^2}\right), \tag{18}$$

and C_1–C_4 are constants defined in Appendix A, $\forall\gamma \in (0,1), z = 1, 2, \ldots$.

Proof. See Appendix A for the complete proof. □

Pleae note that when $t = (k-1)\tau, \forall[k]$, we have $\|\boldsymbol{x}_{+}^t - \boldsymbol{x}_{[k]}^t\| = 0 = h(0)$, which also satisfises (18). We note that $F(\boldsymbol{x})$ is ρ-Lipschitz, so we also have

$$F(\boldsymbol{x}^t) - F(\boldsymbol{x}_{[k]}^t) \le \rho f(t-(k-1)\tau). \tag{19}$$

We then bound the gap between $\boldsymbol{x}_{+}^t$ and $\boldsymbol{x}^t$ in Theorem 2.

Theorem 2. *For any interval $[k]$, suppose $\gamma \in (0,1), s_i^k \in [0,1]$, and $\pi \in [0,1]$, we have*

$$\|\boldsymbol{x}_+^{k\tau} - \boldsymbol{x}^{k\tau}\| \leq 2\mu\eta\rho q^k \pi, \tag{20}$$

where we define $q^k \triangleq \sum_{i=1}^N \frac{D_i}{D} s_i^k$ as the weighted average of all workers' scores at k's aggregation, and constant μ is defined in Appendix B.

Proof. See Appendix B for the complete proof. □

Theorem 3. *Suppose (1) $\beta\eta(\gamma+1) \in (0,1]$, $\gamma \in (0,1)$, and $\forall \tau = 1,2,\ldots$; (2) $\omega\alpha\sigma^2 - \frac{\rho(f(\tau)+2\mu\eta\rho\varphi\pi)}{\tau\varepsilon^2} > 0$; (3) $F(\boldsymbol{x}_{[k]}^{k\tau}) - F(\boldsymbol{x}^*) \geq \varepsilon, \forall k$; and (4) $F(\boldsymbol{x}_+^T) - F(\boldsymbol{x}^*) \geq \varepsilon$ are satisfied. $\exists \varepsilon > 0$, the upper bound of Algorithm 1 is given by*

$$F(\boldsymbol{x}_+^T) - F(\boldsymbol{x}^*) \leq \frac{1}{T\left(\omega\alpha\sigma^2 - \frac{\rho(f(\tau)+2\mu\eta\rho\varphi\pi)}{\tau\varepsilon^2}\right)}. \tag{21}$$

We define $F(\boldsymbol{x}^*)$ as the minimum value, if there exists some $\zeta > 0$ such that $F(\boldsymbol{x}^*) \leq F(\boldsymbol{x})$ for all x within distance ζ of $\boldsymbol{x}^*$. Constants ω, σ, φ and α are defined in Appendix C.

Proof. See Appendix C for the complete proof. □

Theorem 3 demonstrates that Algorithm 1 converges with the convergence rate $\mathcal{O}\left(\frac{1}{T}\right)$ for smooth non-convex problems under non-i.i.d. data distribution. The overall gap $F(\boldsymbol{x}_g^T) - F(\boldsymbol{x}^*)$ decreases when T is larger. From [13, Appendix C], we have $f(z) \geq 0$ for any $z = 1,2,\ldots,$ and it increases with z. Therefore, the value of $\frac{\rho(f(\tau)+2\mu\eta\rho\varphi\pi)}{\tau\varepsilon^2}$ increases with τ so as to increase the overall bound. However, in order to let the condition 2 in Theorem 3 hold, we cannot set a very large τ, implying that convergence is guaranteed when $f(\tau)$ is below a certain threshold. Meanwhile, condition 2 in Theorem 3 holds when ε is above a positive certain threshold defined as ε_0. When ε is small and $\varepsilon > \varepsilon_0$, i.e., when the loss function is close to the stationary point, the value of $\omega\alpha\sigma^2 - \frac{\rho(f(\tau)+2\mu\eta\rho\varphi\pi)}{\tau\varepsilon^2}$ diminishes, resulting in an increased overall bound. In this case, a larger T is required to achieve convergence towards a small ε. Conversely, the algorithm requires fewer iterations (smaller T) to only converge to a large ε. This illustrates the trade-off between the number of training iterations and the tightness of the bound.

5 Experimental Results

In this section, we evaluate the convergence performance of pFedMo with various mainstream momentum-based algorithms, including FastSlowMo [15], DOMO [16], FedADC [17], FedMom [11], SlowMo [12], FedNAG [13], Mime [14], FedProx [18], and classical FL algorithm FedAvg [5]. We compare pFedMo with benchmarks in two different environments, considering two aspects. ① Given the same total training iterations T, we compare the accuracy in an GPU tower server. ② Given an accuracy goal, e.g., 85%, we compare the total training time in a real-world IoT system. We then evaluate the effects of hyper-parameters such as τ, γ, π, and N. Afterwards, we manually establish three distinct non-i.i.d. data distribution scenarios to evaluate the effects of non-i.i.d. data. 1) We assign a subset of classes out of total classes of dataset to each worker, referring to as "Label-Skew non-i.i.d.". 2) We assign a distinct quantity of data samples to each worker, referring to as "Quantity-Skew non-i.i.d.". 3) We utilize Dirichlet distribution [44] to combine 1) and 2), letting Label-Skew and Quantity-Skew occur simultaneously.

5.1 Experiment on Convergence of pFedMo

5.1.1 Experimental Setup

We use a GPU tower server with 4 NVIDIA GeForce RTX 2080Ti GPUs to test the convergence performance. Two machine learing tasks are conducted base on four real-world datasets. ① Image classification: MNIST [19], CIFAR-10 [20], and ImageNet [21], [45]. ② Human activity recognition: UCI-HAR [22]. For the MNIST dataset, it contains 60,000 training images, 10,000 test images for 10 different class. The CIFAR-10 dataset contains 50,000 images for training and 10,000 images for test over 10 classes. The images in MNIST are 28×28 gray-scaled dimensional. The images in CIFAR-10 are $32 \times 32 \times 3$ dimensional. The ImageNet dataset is a tiny version and is more challenging for training the model for image classification tasks, as it contains 200 different classes. For each class, it has 500 images. Overall, there is 100,000 image for training and 10,000 images for testing. The images in Tiny-ImageNet are $64 \times 64 \times 3$ dimensional. The UCI-HAR dataset comprises 10,299 instances characterized by 561 attributes, distributed across six distinct activities. We use five models including Linear Regression, Logistic Regression, LeNet5, VGG16, and ResNet18. The first two models are convex models which are widely used in segmentation task. LeNet5 is a classic CNN model [24]. VGG16 and ResNet18 are DNN models with the structure defined in [21], [25] respectively. All training and testing data are randomly shuffled and distributed among workers. We use mini-batch with size 64 in all experiments. The learning rate η is set to 0.01. The specific hyper-parameters setting will be defined in each experiment accordingly.

5.1.2 Performance Comparison

In Table 2, we compare the convergence performance of pFedMo with benchmark algorithms based on various models and datasets. The values in the table illustrate the training accuracy for different algorithms under a given T training iterations. The higher accuracy indicates the better performance. We set $T = 1000$ (MNIST), $T = 4000$ (UCI-HAR), and $T = 10000$ (CIFAR-10 and ImageNet), $\tau = 20$ (convex models) and $\tau = 40$ (non-convex models), $\gamma = 0.5$, $N = 4$, and $\pi = 1$. We categorize the benchmarks into three categories: ① double-momentum implemented on both workers and the aggregator (FastSlowMo, DOMO, and FedADC), ② single-momentum implemented on either worker or the aggregator (FedMom, SlowMo, FedNAG, and Mime), and ③ no-momentum (FedProx and FedAvg). For convenient presentation, we use "$>$" to indicate "is better than".

First, we observe that pFedMo $>$ all benchmarks. For convex models, pFedMo increases the training accuracy by 0.21–2.47%. For non-convex models, pFedMo achieves the accuracy increase by 1.00–35.90% (CNN) and 0.50–8.12% (DNN), respectively. This demonstrates that applying personalized momentum on both workers and the aggregator

TABLE 2
Performance comparison of different FL algorithms (accuracy %).

	Linear on MNIST	Logistic on MNIST	LeNet5 on MNIST	LeNet5 on CIFAR10	VGG16 on CIFAR10	ResNet18 on ImageNet	LeNet5 on UCI-HAR
pFedMo	**86.00** $\pm$ 0.06	**89.36** $\pm$ 0.05	**97.23** $\pm$ 0.03	**64.49** $\pm$ 0.11	**89.88** $\pm$ 0.10	**68.42** $\pm$ 0.09	**89.21** $\pm$ 0.09
FastSlowMo [15]	85.79 $\pm$ 0.05	89.02 $\pm$ 0.05	95.90 $\pm$ 0.05	59.39 $\pm$ 0.07	88.53 $\pm$ 0.09	67.05 $\pm$ 0.10	88.15 $\pm$ 0.06
DOMO [16]	83.74 $\pm$ 0.06	88.83 $\pm$ 0.05	95.59 $\pm$ 0.07	63.49 $\pm$ 0.09	89.16 $\pm$ 0.10	67.58 $\pm$ 0.15	86.05 $\pm$ 0.12
FedADC [17]	85.51 $\pm$ 0.04	88.18 $\pm$ 0.05	95.09 $\pm$ 0.07	56.00 $\pm$ 0.11	89.38 $\pm$ 0.08	67.76 $\pm$ 0.12	85.14 $\pm$ 0.09
FedMom [11]	84.84 $\pm$ 0.06	88.05 $\pm$ 0.05	94.74 $\pm$ 0.05	54.87 $\pm$ 0.07	88.03 $\pm$ 0.10	66.91 $\pm$ 0.11	84.69 $\pm$ 0.07
SlowMo [12]	84.82 $\pm$ 0.06	88.00 $\pm$ 0.06	94.88 $\pm$ 0.05	54.43 $\pm$ 0.06	88.47 $\pm$ 0.09	66.84 $\pm$ 0.09	83.03 $\pm$ 0.10
FedNAG [13]	84.97 $\pm$ 0.04	88.14 $\pm$ 0.05	95.04 $\pm$ 0.06	55.54 $\pm$ 0.09	88.33 $\pm$ 0.06	66.81 $\pm$ 0.14	84.69 $\pm$ 0.06
Mime [14]	84.41 $\pm$ 0.06	87.73 $\pm$ 0.06	93.89 $\pm$ 0.08	48.24 $\pm$ 0.15	81.76 $\pm$ 0.11	64.33 $\pm$ 0.21	76.75 $\pm$ 0.11
FedProx [18]	85.49 $\pm$ 0.05	86.85 $\pm$ 0.05	93.78 $\pm$ 0.06	46.73 $\pm$ 0.12	87.43 $\pm$ 0.05	66.64 $\pm$ 0.10	56.95 $\pm$ 0.07
FedAvg [5]	83.57 $\pm$ 0.04	86.89 $\pm$ 0.05	93.31 $\pm$ 0.08	37.79 $\pm$ 0.19	88.27 $\pm$ 0.15	66.59 $\pm$ 0.09	53.31 $\pm$ 0.12

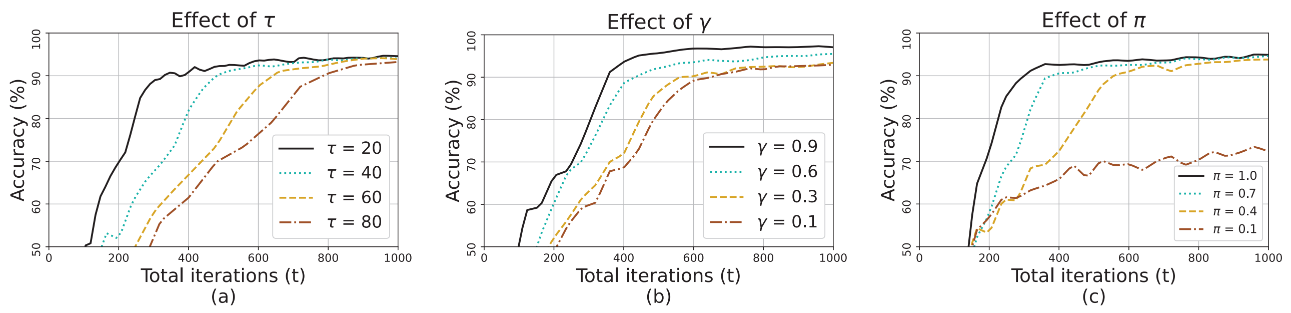


Fig. 2. Accuracy comparison for pFedMo under different settings of τ, γ and π.

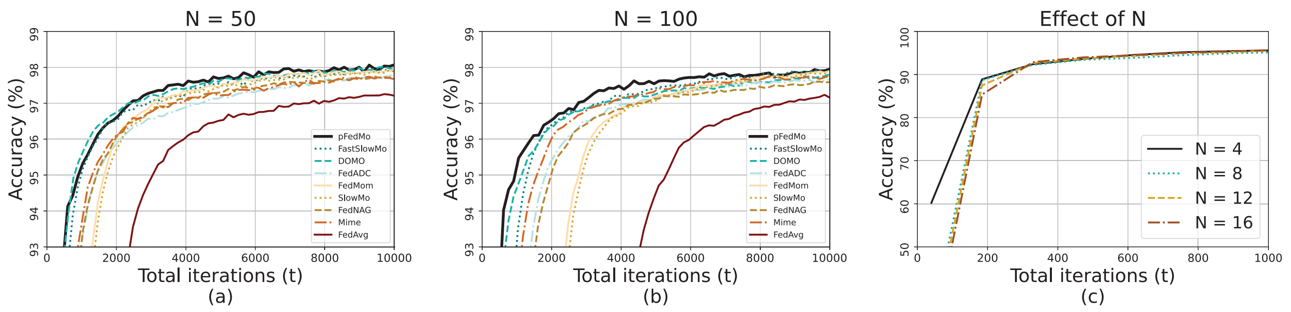


Fig. 3. (a) and (b): Accuracy comparison for pFedMo and benchmarks for more workers ($N = 50$ and $N = 100$). (c) Accuracy comparison for pFedMo with different number of workers ($N = 4, 8, 12, 16$)

as well as the personalized model achieves the best performance compared to the algorithms without personalization. It is very useful to evaluate performance by considering various factors such as the diversity of models and datasets, as well as a wide range of SOTA FL algorithms, because performance varies based on these factors. Compared to five models, four datasets, and nine SOTA benchmarks, the result in Table 2 illustrates that pFedMo achieves a sufficient and substantial performance improvement.

Second, we observe that ① > ② > ③, with 0.04–15.25% accuracy increase from ② to ①, and 0.11-31.38% accuracy increase from ③ to ②. This verifies that the momentum accelerates the convergence compared to FedProx and FedAvg without momentum. Specifically, double-momentum algorithms outperform single-momentum algorithms.

Third, we observe that FedProx outperforms FedAvg in most cases, which illustrates that the model personalization accelerates the convergence.

5.1.3 Effects of Hyper-parameters

To understand how different hyper-parameters affect the convergence performance of pFedMo, we conduct several experiments using LeNet5 on MNIST dataset to analyze the effects of aggregation period τ, momentum factor γ, and personalization temperature π. Only 3 classes of data (*3-class non-i.i.d.*) are distributed among workers.

Effects of τ: In Fig. 2(a), we evaluate the effects of τ. We set $T = 1000, \gamma = 0.5, \pi = 0.5, N = 4$. We observe that large τ decreases the training performance especially at the early stage of the training. This verifies the result of Theorem 1. The large τ increases the value of $f(\cdot)$, and thus increases the overall bound. As a result, the training performance is decreased.

Effects of γ: In Fig. 2(b), we evaluate the effects of γ. We set $T = 1000, \tau = 40, \pi = 0.5, N = 4$. We observe that

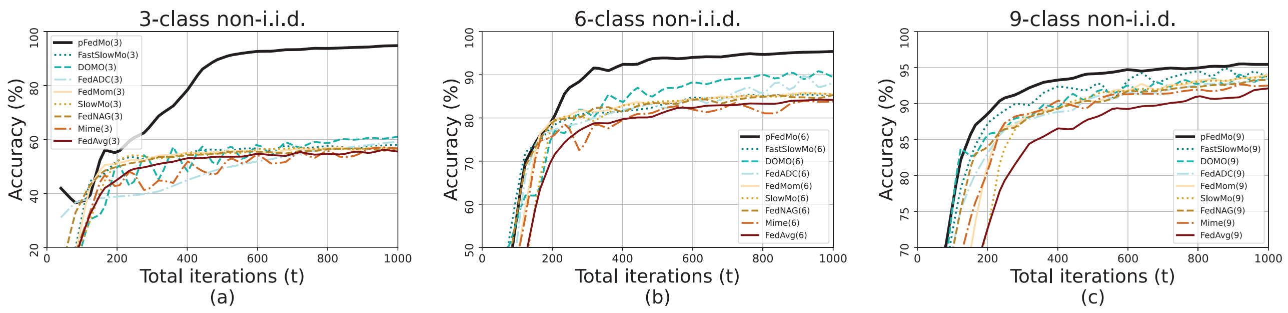


Fig. 4. Accuracy comparison for pFedMo under 3-class (a), 6-class (b), and 9-class (c) Label-Skew non-i.i.d. data.

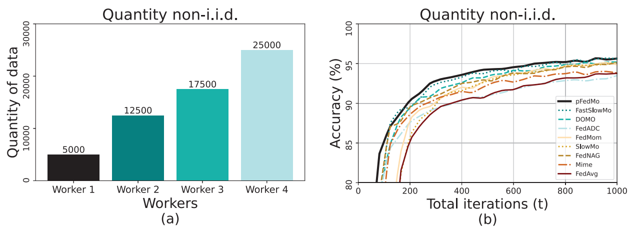


Fig. 5. Accuracy comparison of pFedMo with other benchmarks under Quantity-Skew non-i.i.d data.

large γ increases the training performance. This matches our expectation. Momentum accelerates the convergence by comparing the difference between past and current model vectors. If we set the large momentum factor γ (but should < 1), the acceleration performance further improves. Such observation is also consistent with those in [13], [15].

Effects of π: In Fig. 2(c), we evaluate the effects of π. We set $T = 1000, \tau = 40, \gamma = 0.5, N = 4$. We observe that large π increases the training performance. From (11) and (12), we can see that large π increases the weight (contribution) of the representation momentum and model at workers. That is to say, large π accelerates the distillation of knowledge from the representation model to worker models, thereby enhancing the training performance.

Effects of N: To emulate the cross-silo FL [40] (typically up to 100 participants), we compare the training accuracy when more workers are involved in the training ($N = 50$ and $N = 100$). In Fig. 3(a) and (b), although large N decreases the performance for all algorithms (Large N causes more data divergence among workers.), we can still observe that pFedMo achieves the highest accuracy and the results show the same trend as in Table 2. In Fig. 3 (c), we evaluate the effects of the number of workers N. We set $T = 1000, \tau = 40, \gamma = 0.5, N = 4, 8, 12, 16$. We observe the large N will cause a decline in the convergence performance, especially in the early stage of training. This follows our expectation because more workers cause more divergence among the workers and thus decrease convergence performance. After a sufficient number of training iterations, we can still observe that the accuracy of more worker cases will be close to the few worker cases when they finally converge.

5.1.4 Effects of Non-i.i.d. Data

To analyze how non-i.i.d. data affects the convergence performance of pFedMo and benchmarks, we conduct several experiments using LeNet5 on MNIST dataset based on three non-i.i.d. data distribution scenarios (Label-Skew, Quantity-Skew, and combination of them). The setting is $T = 1000, \gamma = 0.5, \pi = 0.5, N = 4$.

For Label-Skew non-i.i.d., we randomly assign a subset of classes, selected from the total 10 classes in MNIST dataset. For fair comparison, each worker is allocated with the same number of data samples. Smaller x represents a higher level of Label-Skew non-i.i.d. setting. We use *3-class non-i.i.d.*, *6-class non-i.i.d.*, and *9-class non-i.i.d.* to represent *high*, *middle* and *low* level of Label-Skew non-i.i.d. data respectively. In Fig. 4, we observe that pFedMo > DOMO ≈ FastSlowMo > FedADC > FedNAG > FedMom > SlowMo > Mime ≈ FedAvg, which is consistent with the result in Table 2. The pFedMo algorithm achieves at least 34.26%, 5.94%, and 1.28% accuracy increase for *high*, *middle*, and *low* level of Label-Skew non-i.i.d. setting respectively. This demonstrates that pFedMo outperforms benchmarks under any level of Label-Skew non-i.i.d. data distribution. We also observe that pFedMo is more robust to different levels of Label-Skew non-i.i.d. data with only 0.17% drop from *low* to *high*, while the performance of benchmarks declines fast (33.15–37.45% drop from *low* to *high*).

For Quantity-Skew non-i.i.d., we assign an exact number of data samples to workers as shown in Fig. 5(a) (Worker 1: 5000, Worker 2: 12500, Worker 3: 17500, and Worker 4: 25000). For fair comparison, each worker contains all 10 classes with each class containing the same portion of data samples. Please note that the typical aggregation method in FL [5] employs the weighted average, i.e., the weight for each worker model is proportional to the number of its data samples. Nevertheless, in order to evaluate the effect of Quantity-Skew non-i.i.d., we assign the same weight (0.25) for all four workers specifically in this experiment. In Fig. 5(b), we still observe that pFedMo outperforms benchmarks, with the 95.71% accuracy.

We further conduct a more complicated case where Label-Skew and Quantity-Skew non-i.i.d. occur at the same time. To achieve this, we employ the Dirichlet distribution [44], denoted as $Dir(\xi)$, ensuring variation in both classes and data sample quantities among workers. $\xi (> 0)$ is a hyper-parameter that controls the level of non-i.i.d., where a small ξ indicates a higher level of heterogeneous (non-

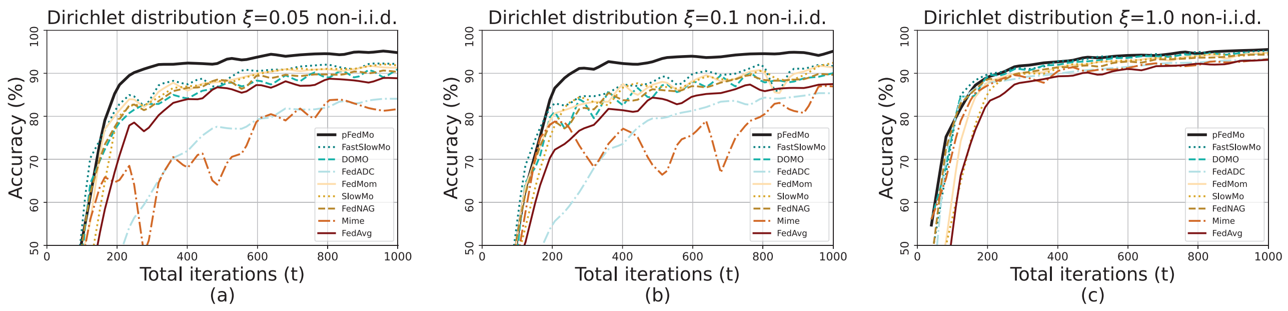


Fig. 6. Accuracy comparison for pFedMo under Dirichlet distribution $\xi = 0.05$ (a), $\xi = 0.1$ (b), and $\xi = 1.0$ (c).

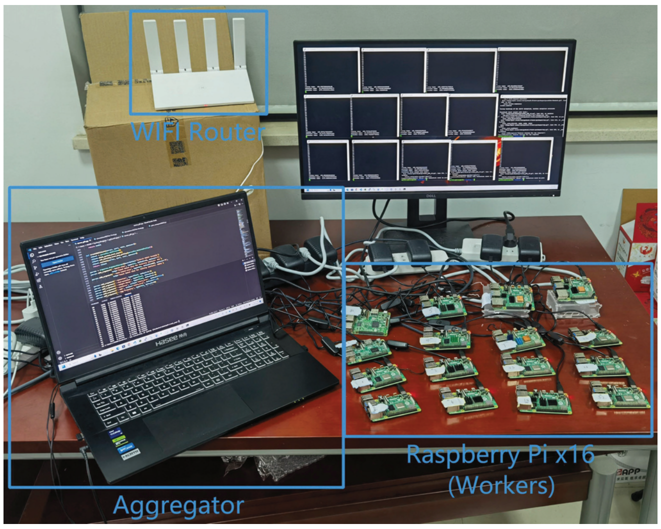


Fig. 7. System Architecture of Real-world IoT system in the experiment.

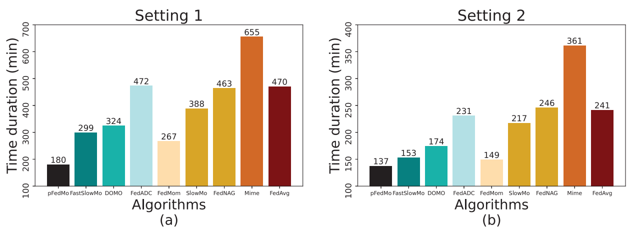


Fig. 8. Comparison of total training time to reach 85% accuracy under two different settings for fire detection application. The time is labeled above each bar. (a): $\gamma = 0.5, \pi = 1, \tau = 20$. (b): $\gamma = 0.5, \pi = 1, \tau = 40$.

i.i.d.) data. In this experiment, we set $\xi = \{0.05, 0.1, 1\}$ to simulate *high*, *middle*, and *low* levels of non-i.i.d. data distribution respectively[1]. In Fig. 6, we observe the same results as shown in Figs. 4 and 5. The pFedMo algorithm achieves 95.54%, 95.13%, and 94.8% accuracy in the *low*, *middle*, and *high* level of non-i.i.d. settings, demonstrating that pFedMo is more robust compared to benchmarks, evidenced by a mere 0.74% accuracy drop from *low* to *high*.

1. In practice, setting $\xi \geq 1$ generates an approximate i.i.d. data distribution (low level of non-i.i.d.), which has widely been used in the literature [46], [47]. Please refer to [48] for more details.

5.2 Experiement on Real-world IoT System

To evaluate the performance of pFedMo and benchmarks in a more realistic environment, we build up a real-world IoT Edge Computing system which runs fire detection application. We use 16 Raspberry Pi 4 Model B as workers and one laptop (HASEE G9R9 with Intel Core i9-13900H CPU) as the aggregator. Workers and the aggregator are connected to HUAWEI WS5200 router with 2.4GHz WIFI and 1Gbps Ethernet wired cable, respectively. The IoT system architecture is illustrated in Fig. 7. The total training time is calculated from the moment the aggregator initiates the training process for all workers until the aggregator determines that the training accuracy has reached the specified goal (e.g., 85%). The application runs FireNet [26] (a modified version of AlexNet [49]) to detect fire present in captured pictures or video frames. The dataset can be found in [23].

In Fig. 8, we observe that under two different settings ① $\gamma = 0.5, \pi = 1, \tau = 20$ and ② $\gamma = 0.5, \pi = 1, \tau = 40$, to reach accuracy goal 85%, pFedMo spends 180min for setting ① and 137min for setting ②. Benchmarks spend 267–655min for setting ① and 149–361min for setting ②. This demonstrates that pFedMo is superior than benchmarks and achieves the 1.09–3.64x training time speedup compared to benchmarks in different scenarios.

6 Conclusion

In this paper, we propose pFedMo, a personalized Federated Learning with contrastive momentum algorithm. A score function is designed to personalize worker momentum and model at each aggregation phrase. We provide convergence analysis of pFedMo with a convergence rate of $\mathcal{O}\left(\frac{1}{T}\right)$ for smooth non-convex problems under non-i.i.d. data. We experimentally evaluate the convergence performance of pFedMo compared to mainstream momentum-based algorithms without personalization from the perspectives of training accuracy and training time. It empirically shows that pFedMo consistently improves the convergence performance especially on highly heterogeneous (non-i.i.d.) data.

References

[1] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of cleaner production*, vol. 140, pp. 1454–1464, 2017.

[2] A. Solanas, C. Patsakis, M. Conti, I. S. Vlachos, V. Ramos, F. Falcone, O. Postolache, P. A. Pérez-Martínez, R. Di Pietro, D. N. Perrea *et al.*, "Smart health: A context-aware health paradigm within smart cities," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 74–81, 2014.

[3] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, "A review of machine learning and iot in smart transportation," *Future Internet*, vol. 11, no. 4, p. 94, 2019.

[4] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[5] B. McMahan, E. Moore, D. Ramage, and etc., "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[6] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

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

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[10] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[11] Z. Huo, Q. Yang, B. Gu, L. C. Huang *et al.*, "Faster on-device training using new federated momentum algorithm," *arXiv preprint arXiv:2002.02090*, 2020.

[12] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "SlowMo: Improving communication-efficient distributed sgd with slow momentum," in *International Conference on Learning Representations*, 2020.

[13] Z. Yang, W. Bao, D. Yuan, N. H. Tran, and A. Y. Zomaya, "Federated learning with nesterov accelerated gradient," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4863–4873, 2022.

[14] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Mime: Mimicking centralized stochastic algorithms in federated learning," *arXiv preprint arXiv:2008.03606*, 2020.

[15] Z. Yang, S. Fu, W. Bao, D. Yuan, and A. Y. Zomaya, "FastSlowMo: Federated learning with combined worker and aggregator momenta," *IEEE Transactions on Artificial Intelligence*, 2022.

[16] A. Xu and H. Huang, "Coordinating momenta for cross-silo federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8735–8743.

[17] E. Ozfatura, K. Ozfatura, and D. Gündüz, "FedADC: Accelerated federated learning with drift control," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE Press, 2021, p. 467–472.

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

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," *N/A*, 2009.

[21] T. Moon and T. Ryffel, *Pytorch-Tiny-ImageNet*, 6 2020. [Online]. Available: https://github.com/tjmoon0104/pytorch-tiny-imagenet

[22] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proceedings of the 21th international European symposium on artificial neural networks, computational intelligence and machine learning*, 2013, pp. 437–442.

[23] A. Dunnings and T. Breckon, *Fire Detection Datasets*, 8 2020. [Online]. Available: https://github.com/tobybreckon/fire-detection-cnn/blob/master/README.md

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] S. Gross, S. Chintala, N. Hug, L. Yeager, and E. R. etc., *Pytorch-VGG*, 5 2021. [Online]. Available: https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py

[26] A. Dunnings and T. Breckon, "Experimentally defined convolutional nerual network architecture variants for non-temporal real-time fire detection," in *Proc. International Conference on Image Processing*. IEEE, 9 2018, pp. 1558–1562.

[27] C. T. Dinh, N. H. Tran, T. D. Nguyen, W. Bao, A. Y. Zomaya, and B. B. Zhou, "Federated learning with proximal stochastic variance reduced gradient algorithms," in *Proceedings of the 49th International Conference on Parallel Processing*, 2020, pp. 1–11.

[28] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[29] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[31] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

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

[33] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[34] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," *arXiv preprint arXiv:1906.06268*, 2019.

[35] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[36] G. Goh, "Why momentum really works," *Distill*, 2017. [Online]. Available: http://distill.pub/2017/momentum

[37] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)," *Doklady ANSSSR (translated as Soviet.Math.Docl.)*, vol. 269, pp. 543–547, 1983.

[38] S. Bubeck, "Convex optimization: Algorithms and complexity," *arXiv preprint arXiv:1405.4980*, 2014.

[39] Y. Yan, T. Yang, Z. Li, Q. Lin, and Y. Yang, "A unified analysis of stochastic momentum methods for deep learning," in *IJCAI*, 2018, pp. 2955–2961.

[40] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[41] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[42] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[43] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE JSAC*, vol. 37, no. 6, pp. 1205–1221, 2019.

[44] T. Minka, "Estimating a dirichlet distribution," 01 2003.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

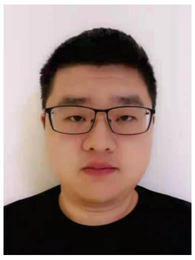
[46] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021, pp. 12 878–12 889.

[47] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," vol. 33, 2020, p. 2351–2363.

[48] D. J. MacKay, *Information Theory, Inference and Learning Algorithms.* Cambridge University Press. [Online]. Available: http://www.inference.org.uk/mackay/itila/

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[50] I. Mitliagkas and J. Gallego, "Ift 6085: Theoretical principles for deep learning," in *University of Montreal*. University of Montreal, 2021. [Online]. Available: http://mitliagkas.github.io/ift6085-dl-theory-class/



Sen Fu received the Bachelor of Computer Science and Technology (Advanced) (Honours) degree in the University of Sydney in 2021. He is currently a PhD student at the School of Computer Science, the University of Sydney. His research focuses on the distributed machine learning.

Zhengjie Yang received the Master of Information Technology degree in the University of Sydney in 2017. In 2017–2018, he worked as a software engineer in Garvan Institute of Medical Research and Link Group Pty Ltd in Sydney. In 2019, he started pursuing the PhD degree in the School of Computer Science, the University of Sydney. His research focuses on the edge computing and distributed machine learning.

Chuang Hu received his B.S and M.S. degrees in Computer Science from Wuhan University in 2013 and 2016, and Ph.D. degree from the Hong Kong Polytechnic University in 2019. He is an Associate Researcher in the School of Computer Science at Wuhan University. His research interests include edge learning, federated learning/analytics, and distributed computing.

Wei Bao (S'10–M'16) received the B.E. degree in Communications Engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009; the M.A.Sc. degree in Electrical and Computer Engineering from the University of British Columbia, Vancouver, Canada, in 2011; and the PhD degree in Electrical and Computer Engineering from the University of Toronto, Toronto, Canada, in 2016. He is currently a senior lecturer at the School of Computer Science, the University of Sydney, Sydney, Australia. His research covers the area of network science, with particular emphasis on Internet of things, mobile computing, edge computing, and distributed learning. He received the Best Paper Awards in ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) in 2013 and 2019 and IEEE International Symposium on Network Computing and Applications (NCA) in 2016.